Load-optimization in Reconfigurable Data-center Networks: Algorithms and Complexity of Flow Routing

WENKAI DAI, Faculty of Computer Science, University of Vienna, Austria
KLAUS-TYCHO FOERSTER, Faculty of Computer Science, TU Dortmund University, Germany
DAVID FUCHSSTEINER, Faculty of Computer Science, University of Vienna, Austria
STEFAN SCHMID, TU Berlin, Germany

Emerging reconfigurable data centers introduce unprecedented flexibility in how the physical layer can be programmed to adapt to current traffic demands. These reconfigurable topologies are commonly hybrid, consisting of static and reconfigurable links, enabled by e.g., an Optical Circuit Switch (OCS) connected to top-of-rack switches in Clos networks. Even though prior work has showcased the practical benefits of hybrid networks, several crucial performance aspects are not well understood. For example, many systems enforce artificial segregation of the hybrid network parts, leaving money on the table.

In this article, we study the algorithmic problem of how to jointly optimize topology and routing in reconfigurable data centers, in order to optimize a most fundamental metric, maximum link load. The complexity of reconfiguration mechanisms in this space is unexplored at large, especially for the following cross-layer network-design problem: given a hybrid network and a traffic matrix, jointly design the physical layer and the flow routing in order to minimize the maximum link load.

We chart the corresponding algorithmic landscape in our work, investigating both un-/splittable flows and (non-)segregated routing policies. A topological complexity classification of the problem reveals NP-hardness in general for network topologies that are trees of depth at least two, in contrast to the tractability on trees of depth one. We moreover prove that the problem is not submodular for all these routing policies, even in multi-layer trees.

However, networks that can be abstracted by a single packet switch (e.g., nonblocking Fat-Tree topologies) can be optimized efficiently, and we present optimal polynomial-time algorithms accordingly. We complement our theoretical results with trace-driven simulation studies, where our algorithms can significantly improve the network load in comparison to the state-of-the-art.

CCS Concepts: • **Networks** → **Network architectures**; • **Theory of computation** → **Design and analysis of algorithms;**

Additional Key Words and Phrases: Network design, load optimization, optical networks, algorithms, complexity

Research supported by the European Research Council (ERC), grant agreement No. 864228 (project AdjustNet).
Authors' addresses: W. Dai and D. Fuchssteiner, Faculty of Computer Science, University of Vienna, Waehringer Str. 29, Vienna, Vienna, 1090, Austria; emails: {wenkai.dai, david.alexander.fuchssteiner}@univie.ac.at; K.-T. Foerster, Faculty of Computer Science, TU Dortmund University, Otto-Hahn-Str. 16, Dortmund, North Rhine-Westphalia, 44227, Germany; email: klaus-tycho.foerster@tu-dortmund.de; S. Schmid, TU Berlin, Einsteinufer 17, Berlin, Berlin, 10587, Germany; email: stefan.schmid@tu-berlin.de.

2376-3639/2023/07-ART8 $15.00
https://doi.org/10.1145/3597200

ACM Reference format:
Wenkai Dai, Klaus-Tycho Foerster, David Fuchssteiner, and Stefan Schmid. 2023. Load-optimization in Reconfigurable Data-center Networks: Algorithms and Complexity of Flow Routing. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 8, 3, Article 8 (July 2023), 30 pages.
https://doi.org/10.1145/3597200

1 INTRODUCTION

Data centers nowadays empower everyday life in aspects such as business, health, and industry, but also science and social interactions. With the rise of related data-intensive workloads as generated by machine learning, artificial intelligence, and the distributed processing of big data in general, data center traffic is growing very fast [63, 73]. Much of this traffic is internal to data centers, evoking considerable interest in data center design problems [64, 84].

Herein the emergence of a *programmable physical layer*, enabled by optical circuit switches [29, 50, 82], free-space optics [12, 37], or beamformed wireless connections [44, 45], leads to intriguing new possibilities, as leveraging fully electrically packet switched networks "*is increasingly cost prohibitive and likely soon infeasible*" [60, 62], see also the recent report by Microsoft [27]. In other words, electrical chips are unlikely to deliver sufficient performance for next-generation networks, and in turn, we must rely on programmable optical topologies for increased bandwidth, connectivity, and power-efficiency [5].

Extensive past work has already shown significant benefits of such reconfigurable data center networks [34, 43], but the underlying complexity is not well understood [11]. For example, many works artificially restrict their flow routing policies to be segregated between programmable and static network parts, aiming to place elephant flows on reconfigurable links [33].

Whereas some general algorithmic results exist w.r.t. latency [32, 37] or specific traffic patterns [10, 80], complexity questions of network-design for the objective of load-optimization are mostly uncharted. The exceptions are the work by Yang et al. [87], which focuses on the hardness induced by wireless interference, the work by Zheng et al. [91], who provide intractability results on general non-data-center topologies, and the results by Dai et al. [20], which uncover the approximation hardness for special settings. However, tree-induced topologies, as commonly employed in data centers, e.g., Fat-Tree, have yet not been exposed to a fine-grained complexity analysis, which can reveal a complexity dichotomy between network designs, as we will show in this article.

At the same time, link load is a most central performance metric [15, 44, 46, 76], and flow routing in traditional networks has been investigated for decades already [3]. We are thus motivated by the desire to take the first steps towards fundamentally understanding the network-design problem for load-optimization in data center networks, jointly considering flow routing and (interference-free) physical layer programmability enabled by, e.g., optical circuit switches.

1.1 Contributions

This article initiates the network-design study of load-optimization in reconfigurable networks with optical circuit switches, leveraging the flexibility of emerging programmable physical layers for flow routing. We investigate multiple problem dimensions, from splittable to unsplittable flows, to fully flexible (non-segregated) versus segregated routing policies. Our results not only include efficient algorithms and complexity characterizations but also simulations on real-world workloads:

(1) *Complexity:* We prove strong NP-hardness for non-segregated and segregated routing on tree networks of height greater or equal than two, for un-/splittable flow models, excluding

Table 1. Network-design Complexity for Load-optimization in Reconfigurable Networks for Un-/splittable and Non-/segregated Routings when the Topologies are Trees of Height $h = 1$ and $h \geq 2$

Time Complexity	Splittable	Segregated	Height of Tree h	References
Strongly NP-hard	Yes	Yes	$h \geq 2$	Theorem 3.2
Strongly NP-hard	Yes	No	$h \geq 2$	Theorem 3.6
Strongly NP-hard	No	Yes	$h \geq 2$	Theorem 3.2
Strongly NP-hard	No	No	$h \geq 2$	Theorem 3.5
Polynomial-time	Yes	Yes	$h = 1$	Theorem 4.6
Polynomial-time	Yes	No	$h = 1$	Theorem 4.6
Polynomial-time	No	Yes	$h = 1$	Theorem 4.6
Weakly NP-hard	No	Yes	$h = 1$	Theorem 3.4

star networks, which are summarized in Table 1. Moreover, all four problem settings are not submodular w.r.t. load-optimization, preventing common approximation techniques.

(2) *Algorithms:* In turn, we give polynomial-time optimal algorithms for the hybrid switch model of Venkatakrishnan et al. [79], which applies to non-blocking data center interconnects as, e.g., Fat-Trees. To this end, we leverage a combination of subset matching results and topology-specific insights.

(3) *Evaluations:* Our workload-driven simulations (using Facebook, pFabric, and high-performance computing traces) show that our algorithms significantly improve on state-of-the-art methods, decreasing the maximum load by 1.6× to 2.0×.

As a contribution to the research community and in order to ensure the reproducibility of our results, we will make the source code of our algorithms as well as experimental artifacts publicly available at https://gitlab.cs.univie.ac.at/ct-papers/2021-tompecs-load-optimization.

Overview. We start with a formal model and preliminaries in Section 2, followed by complexity (Section 3) results for trees and algorithms for the hybrid switch model (Section 4). We then investigate the performance of our algorithms with trace-driven evaluations in Section 5. Lastly, we discuss related work in Section 6 and conclude in Section 7.

2 MODEL AND PRELIMINARIES

Network model. Let $N = (V, E, \mathcal{E}, C)$ be a *hybrid* network [56, 79] connecting the n nodes $V = \{v_1, \ldots, v_n\}$ (e.g., top-of-the-rack switches), using static links E (usually connected by electrical packet switches). The network N also contains a set of reconfigurable (usually optical) links $\mathcal{E}$. The graph $(V, E \cup \mathcal{E})$ is a bidirected[1] graph such that two directions of each bidirected link $\{v_i, v_j\} \in E$ (respectively, $\{v_i, v_j\} \in \mathcal{E}$), where $v_i, v_j \in V$, work as two (anti-parallel) directed links (v_i, v_j) and (v_j, v_i), respectively. We use the symbol $\overrightarrow{E}$ (respectively, $\overrightarrow{\mathcal{E}}$) to denote the set of corresponding directed links of E (respectively, $\mathcal{E}$). Moreover, a function $C : \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} \mapsto \mathbb{R}^+$ defines capacities for both directions of each bidirected link in $E \cup \mathcal{E}$. Note that $(V, E \cup \mathcal{E})$ can be a multi-graph, e.g., when a reconfigurable link in $\mathcal{E}$ also connects two endpoints of a static link in E.

Reconfigured network. We say that a hybrid network N is *reconfigured* by a reconfigurable switch S if some reconfigurable links $M \subseteq \mathcal{E}$, which must induce *a matching*,[2] are *configured*

[1] Symmetrical connectivity is the standard industry assumption for static cabling, however for reconfigurable links as well. Outside highly experimental hardware, e.g., [37], off-the-shelf products use full-duplex connections [14, 66] and this model assumption is hence prevalent, even in Free-Space Optics [12] proposals.

[2] In other words, no two links in M are adjacent or share an endpoint, enforced by hardware constraints in practice (exclusive connections between ports). We refer to Hecht [47] for an introduction to the technological background.

(implemented) by S to enhance the static network (V, E). The set of configured (bidirected) links M, i.e., a matching, is called *a reconfiguration* of N. The enhanced network obtained by integrating the configured links M with the static links E of the hybrid network N is called a *reconfigured network*, i.e., $N(M) = (V, E \cup M)$. The static network (V, E) of the hybrid network N before reconfiguration can also be thought as a reconfigured network denoted by $N(\emptyset)$.

Hardware. Our results also apply to non-optical switches and links, as long as they match the theoretical properties described in the model. As such, we will only talk about reconfigurable switches and reconfigurable links, implying any appropriate technology that matches our model.

Topologies. Our network model does not place a restriction on the underlying static topology and hence can be applied generally. Notwithstanding, for our hardness results in Section 3, already tree topologies suffice, whereas our positive algorithmic results cover many data center topologies, as we elaborate from Section 4 onwards.

Traffic demands. The resulting network should serve a certain communication pattern, represented as a $|V| \times |V|$ communication matrix $D := (d_{ij})_{|V|\times|V|}$ (*demands*) with non-negative real-valued entries. An entry $d_{ij} \in \mathbb{R}^+$ represents the traffic load (frequency) or a demand from the node v_i to the node v_j. With a slight abuse of notation, let $D(v_i, v_j)$ also denote a demand from v_i to v_j hereafter.

Routing models. For networking, *unsplittable* routing requires that all flows of a demand must be sent along a single (directed) path, while *splittable* routing does not restrict the number of paths used for the traffic of each demand; For a reconfigured network, segregated routing requires flows being transmitted on either static links or configured links, but non-segregated routing admits configured links to be used as shortcuts for flows along static links [29, 82]. Hence, there are *four different routing models*: ***Unsplittable and Segregated (US)***, ***Unsplittable and Non-segregated (UN)***, ***Splittable and Segregated (SS)***, and ***Splittable and Non-segregated (SN)***.

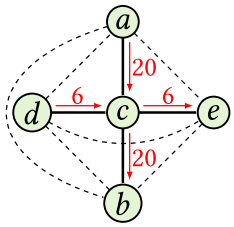
2.1 Load Preliminaries

> "*As minimizing the maximum congestion level of all links is a desirable feature of DCNs [44, 46], the objective of our work is to minimize the maximum link utilization of the entire network.*"
>
> Yang et al. [87], presented at ACM SIGMETRICS 2020 [88]

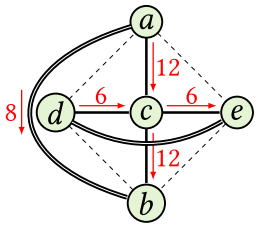
Load optimization. Given a reconfigured network $N(M)$ and demands D, let $f : \overrightarrow{E} \cup \overrightarrow{M} \mapsto \mathbb{R}^+$ be *a feasible flow* serving demands D in $N(M)$ under a routing model $\tau \in \{\text{US}, \text{UN}, \text{SS}, \text{SN}\}$. *The load of each directed link* $e \in \overrightarrow{E} \cup \overrightarrow{M}$ induced by the flow f is defined as $L(f(e)) := f(e)/C(e)$. Then, for a feasible flow f in $N(M)$, *the maximum load* is defined as $L_{\max}(f) := \max\{L(f(e)) : e \in \overrightarrow{E} \cup \overrightarrow{M}\}$, and there must be *an optimal flow* f_{opt} to serve D such that its maximum load is minimized for all feasible flows in $N(M)$. Such an optimal flow is called *a load-optimization flow* in $N(M)$.[3] For a reconfigured network $N(M)$, with a slight abuse of notation, let f^M_{opt} denote an arbitrary load-optimization flow in $N(M)$, then we define a function $L_{\text{min-max}}(N(M)) := L_{\max}(f^M_{\text{opt}})$.

Load-optimization reconfiguration problem. Given a hybrid network N, a routing model $\tau \in \{\text{US}, \text{UN}, \text{SS}, \text{SN}\}$, and demands D, the *τ-load-optimization reconfiguration problem* is to find *an optimal reconfiguration* $M \subseteq \mathcal{E}$ to generate *an optimally reconfigured network* $N(M)$ such that $L_{\text{min-max}}(N(M))$ is minimized for all valid reconfigurations $M_i \subseteq \mathcal{E}$ of N. The *τ-load-optimization*

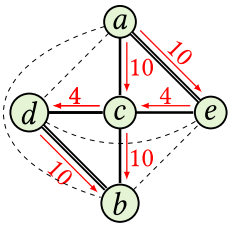
[3] We note that in other works with analogous definitions, load might also be denoted by utilization, and load-optimization by load-balancing.



(a) A hybrid network N without reconfiguration



(b) Reconfiguration (matching): $M_1 = \{\{a, b\}, \{d, e\}\}$



(c) Reconfiguration (matching): $M_2 = \{\{d, b\}, \{a, e\}\}$

Fig. 1. Illustration of a hybrid network N with five nodes $\{a, b, c, d, e\}$ and four static links (drawn solid). A reconfigurable link (dashed) can be created between every pair of nodes, except c, as long as the set of reconfigurable links forms a matching. In this example, the capacity of each link is 20, and the task is to route the five demands $D(a, b) = 8$, $D(a, c) = 6$, $D(c, b) = 6$, $D(d, b) = 6$ and $D(a, e) = 6$, s.t. the maximum link load L is minimized. We utilize the splittable non-segregated (SN) model, which allows arbitrary routing on the (reconfigured) network, and we indicate the flow size on each link except for zero flow. In Figure 1(a), we just consider the static topology, which induces a maximum link load of $20/20 = 1$. A first improvement strategy would be to greedily choose the biggest demand ($D(a, b) = 8$) and create the corresponding reconfigurable link, as in Figure 1(b). Then only a further reconfigurable link between d and e can be created, resulting in a maximum load of $12/20 = 0.6$. In contrast, an optimal solution has a maximum link load of $10/20 = 0.5$, shown in Figure 1(c).

reconfiguration problem is also abbreviated as the τ*-reconfiguration problem* henceforth. We lastly need to find a load-optimization flow for the optimally reconfigured network.

Example. To illustrate the τ-load-optimization reconfiguration problem, we give a small example in Figure 1. Figure 1(a) depicts the hybrid network before adding any reconfiguration, with five nodes $V = \{a, b, c, d, e\}$, four static (bidirected) links E: $\{d, c\}$, $\{b, c\}$, $\{a, c\}$, and $\{e, c\}$ and six reconfigurable (bidirected) links $\mathcal{E}$: $\{a, d\}$, $\{d, b\}$, $\{b, e\}$, $\{a, e\}$, $\{a, b\}$, and $\{d, e\}$.

We consider the routing model $\tau = \text{SN}$ and a capacity function $\forall e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} : C(e) = 20$, with the six demands $D(a, b) = 8$, $D(a, c) = 6$, $D(c, b) = 6$, $D(d, b) = 6$ and $D(a, e) = 6$. In Figure 1(a), each flow can only be routed along static links, creating a link load of $20/20 = 1$ on, e.g., (a, c) with three demands of size 8, 6, 6 from a. In order to improve the maximum link load, one could, e.g., greedily add reconfigurable links in order to reduce the maximum load, such as $\{a, b\}$ in Figure 1(b). Now, the demand $D(a, b) = 8$ is routed directly, reducing the maximum load to just 0.6. Yet, only one further reconfigurable link can be chosen, $\{d, e\}$, without violating the matching constraints. In this situation, any further rerouting does not decrease the maximum link load. For example, when attempting to alleviate the load of 0.6 on (c, b), the load on (a, c) will increase, and vice versa, in the best case canceling each other's load increase.

Notwithstanding, we can improve the maximum load further. To this end, we select $\{a, e\}$ and $\{d, b\}$ as reconfigurable links, as shown in Figure 1(c). At first, this might seem counter-intuitive, as $D(a, e)$ and $D(d, b)$ are only of size 6 each, leaving a load of 0.7 on the links (a, c) and (c, b). However, the demand $D(a, b) = 8$ can be routed indirectly, via the path $\{a, e, c, d, e\}$, yielding an optimal maximum link load of 0.5.

3 COMPLEXITY

In this section, we consider the underlying complexity of the load-optimization problem in reconfigurable networks. We begin with the investigation of NP-hardness, where we study segregated

routing (Section 3.1) and non-segregated routing (Section 3.2). For all four routing models, we prove NP-hardness for trees of any height of two or greater.

Yang et al. [87] considered the case of unsplittable segregated routing on trees and weak NP-hardness, i.e., for large demand sizes. Our NP-hardness results also hold for small demand sizes and we moreover extend the previous result [87] to trees of height one. To show hardness, we can consider special cases, where all directed links have the same capacity of $\gamma \in \mathbb{R}^+$. In particular, we set $\gamma = 1$ in all our NP-hard proofs s.t. the load of each link equals the flow size on itself, but our proofs work for arbitrary γ.

We then prove in Section 3.3 that all four routing models are not submodular, i.e., resist common approximation schemes. Venkatakrishnan et al. [79] considered different objective functions and showed submodularity for the hybrid switching model, resulting in approximation algorithms which therefore cannot be applied here.

3.1 Segregated Routing

We start with the case of segregated routing w.r.t. NP-hardness. The following and some later proofs will make use of the strongly NP-hard 3-Partition problem, which we define first:

Definition 3.1 (3-Partition [36]). Given a finite set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$, and a size function: $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that each $s(a)$ satisfies $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$, can we partition A into m disjoint sets $A_1, \ldots, A_m$, such that for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$, where $|A_i| = 3$?

Theorem 3.2. *The τ-load-optimization reconfiguration problem, where $\tau \in \{US, SS\}$, are strongly NP-hard when the given hybrid network N, before reconfiguration, is a tree of height $h \geq 2$.*

Proof. We first consider the US-load-optimization reconfiguration problem. Given an instance of 3-Partition (A, B, s), we construct an instance of the US-reconfiguration problem as follows: the constructed tree T has the node r as its root, and for each element $a_i \in A$, r has a direct child $s_i \in S$. The root r also has m subtrees T_i for $1 \leq i \leq m$. For each subtree T_i, its root is the node r_i, which is a direct child of r, and the root r_i has $3m$ direct child nodes $F^i = \{f_1^i, \ldots, f_{3m}^i\}$. The tree T represents the static (bidirected) links E and nodes V of the hybrid network N. For the set of all reconfigurable links $\mathcal{E}$, if two nodes $u \in V$ and $v \in V$ are not connected by a static link in T, then there is a reconfigurable (bidirected) link $\{u, v\} \in \mathcal{E}$. Regarding demands D, for each $a_i \in A$, we define $D(f_i^k, s_i) = s(a_i)$ for each $1 \leq k \leq m$. Note that the constructed tree only has a height of two. For each $e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$, the capacity $C(e) = 1$. We claim that after N being reconfigured, there is no directed link $e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$ that has load higher than $(m-1)B$ if and only if there exists a valid 3-Partition for the set A.

Before reconfiguration, each static link $(r_i, r) \in \overrightarrow{E}$ has a flow size of mB. Assume A has a 3-partition $A_1, \ldots, A_m$, where $\sum_{a \in A_i} s(a) = B$ for $1 \leq i \leq m$. For each $A_i = \{a_j, a_k, a_f\}$, where $1 \leq i \leq m$, we connect $s_j, s_k, s_f \in S$ to the corresponding nodes f_j^i, f_k^i, f_f^i in the subtree T_i, by the configured links. Thus, the flow size on each static link $(r_i, r) \in \overrightarrow{E}$ is decreased by B for $1 \leq i \leq m$, i.e., $(m-1)B$.

On the other hand, we assume that we could find a set of configured (bidirected) links $M \subseteq \mathcal{E}$ such that each static (directed) link $(r_i, r) \in \overrightarrow{E}$ for $1 \leq i \leq m$ does not have a flow size more than $(m-1)B$. Note that each element $a_i \in A$ has $B/4 < s(a_i) < B/2$. Due to $D(f_i^k, s_i) = s(a_i)$ for $1 \leq k \leq m$, for any two configured links in M, they cannot convey flows more than B. Moreover, if more than three (bidirected) links are configured between nodes S and the children nodes F^i in the subtree T_i, then there must be another subtree T_j, where $j \neq i$, whose children F^j have at

most two configured links connecting to S, since the set of configured links must be a matching M. Thus, for each subtree T_i, where $1 \le i \le m$, there must be exactly three configured (bidirected) links between three nodes in F^i and three nodes in S. It is known that the direction from F^i to S in each of these three configured links should have a flow of size B. Let $s_j, s_k, s_f \in S$ be these three nodes in S connected to F^i by M, which exactly correspond to a partition $A_i \subseteq A$. Therefore, a valid 3-partition can be obtained.

To prove the hardness of the SS-reconfiguration problem, we use the same construction and claim as for the US-reconfiguration problem above. If A has a 3-partition $A_1, \ldots, A_m$, where $\sum_{a \in A_i} s(a) = B$ for $1 \le i \le m$, then we have proven a valid solution M exists for the US-reconfiguration problem, which is also a solution for SS-reconfiguration since routing for unsplittable flows is a special case of the splittable flow variant. Recall that in the segregated model, a configured (directed) link $(u, v) \in \overrightarrow{\mathcal{E}}$ (respectively, $(v, u) \in \overrightarrow{\mathcal{E}}$) can only carry flows for $D(u, v)$ (respectively, $D(v, u)$). By this setting, we know configured links can only be between leaf nodes of T. If a reconfigurable (bidirected) link $\{s_i, f_i^k\}$, where $k \in \{1, \ldots, m\}$, is configured, then all flows of the demand $D(f_i^k, s_i)$ can go through $(f_i^k, s_i) \in \overrightarrow{\mathcal{E}}$ even under a splittable model due to $D(f_i^k, s_i) \le (m-1)B$. On the other hand, if we can find a set of configured links $M \subseteq \mathcal{E}$ such that each link $(r_i, r) \in \overrightarrow{E}$, where $1 \le i \le m$, does not have a flow size more than $(m-1)B$, then we know each configured (directed) link $(f_i^k, s_i) \in \overrightarrow{M}$ must carry all flows of its demand $D(f_i^k, s_i)$. This corresponds to an unsplittable model, otherwise, there must be a static (directed) link $(r_i, r) \in \overrightarrow{E}$, where $i \in \{1, \ldots, m\}$, which has a flow size more than $(m-1)B$. The same conclusion can be drawn for the SS-reconfiguration problem. □

3.2 Non-segregated Routing

For the non-segregated routing model, we obtain

— weak NP-hardness for trees of height $h = 1$ in the UN model,
— strong NP-hardness for trees of height $h \ge 2$ in the UN model,
— strong NP-hardness for trees of height $h \ge 2$ in the SN model.

For the UN model, we start with the weakly NP-hard case of $h = 1$ in Theorem 3.4, followed by the strongly NP-hard case of $h = 2$ in Theorem 3.5. To show the weakly NP-hardness, we will give a reduction from the weakly NP-hard 2-Partition problem, which is defined as follows:

Definition 3.3 (2-Partition [36]). Given a set of n integers $S = \{s_1, \ldots, s_n\}$ where $B = \sum_{s_i \in S} s_i$, can we divide S into two disjoint subsets S_1 and S_2 such that $\sum_{s_i \in S_1} s_i = \sum_{s_j \in S_2} s_j$?

Theorem 3.4. *The UN-load-optimization reconfiguration problem is weakly NP-hard when the given hybrid network is a hybrid switch network, i.e., a tree of height $h = 1$.*

Proof. We give a reduction from the 2-partition problem, which is weakly NP-hard [36]. Our proof is conceptually similar to the one by Yang et al. [87, Theorem 1], but also applies to the hybrid switch model. For the UN-load-optimization reconfiguration problem, we consider a tree of the height one that has the root node c; for each $s_i \in S$, there is a node $a_i \in A$ connected to c by a static (bidirected) link in E, while c has two additional adjacent nodes r and b. This construction constitutes a hybrid switch network since all nodes only connect to c, where we only allow reconfigurable links between leaf nodes (but not with c). For each $a_i \in A$, we set $D(r, a_i) = s_i$. Without loss of generality, let n be an even number. For each odd number i, where $1 \le i \le n$, we define $D(a_i, a_{i+1}) = B/2$. For capacity, it has $\forall e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} : C(e) = 1$.

After construction, we claim that after N being reconfigured, there is no directed link $e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$ having its load $> B/2$ if and only if there exists a valid 2-Partition for the set A.

Now, the question is how to make links having a load value no more than $B/2$. According to our demands, only one direction in each bidirected link needs to carry traffics. For each odd number i, we need to configure $\{a_i, a_{i+1}\} \in \mathcal{E}$, which gives a matching of $n/2$ bidirected links, otherwise there must be a directed link (c, a_i) that carries flows for demands $D(r, a_i) = B/2$ and $D(r, a_i) = s_i$ for $i \in \{2, \ldots, n\}$. We have to reconfigure the (bidirected) link $\{r, b\} \in \mathcal{E}$, otherwise the load on the link (r, c) is $B = \sum_{s_i \in S} s_i$. Now, for these two directed paths from r to c: $(r, b, c) \in \overrightarrow{\mathcal{E}}$ and (r, c), we have to decide which nodes in A have their flows going through the (directed) link $(r, b) \in \overrightarrow{\mathcal{E}}$ s.t. no (directed) link has load more than $B/2$ in the UN model, which implies a solution to the 2-Partition problem and vice versa. □

Theorem 3.5. *The UN-load-optimization reconfiguration problem is strongly NP-hard when the given hybrid network, before reconfiguration, is a tree of the height $h \geq 2$.*

Proof. We give a reduction from the 3-partition problem. Given an instance of 3-Partition (A, B, s), we construct an instance of the UN-load-optimization reconfiguration problem as follows: the constructed tree T has the node r as its root, and for each element $a_i \in A$, r has a direct child $s_i \in S$. The root r also has m subtrees T_i for $1 \leq i \leq m$. For each subtree T_i, its root is the node r_i, which is a direct child of r, and the root r_i has $3m$ direct child nodes $F^i = \{f^i_1, \ldots, f^i_{3m}\}$. The tree T represents the static (bidirected) links E and nodes V of the hybrid network N. Clearly, the constructed tree only has a height of two. For reconfigurable links $\mathcal{E}$, if two nodes $u \in V$ and $v \in V$ are not connected by a static link in T, then there is a reconfigurable (bidirected) link $\{u, v\} \in \mathcal{E}$. For each $e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$, the capacity $C(e) = 1$. Regarding demands D, for each $a_i \in A$, we define $D(f^k_i, s_i) = s(a_i)$ for each $1 \leq k \leq m$. Moreover, for each node $s_i \in S$, which is a direct child of the root r connected by $\{r, s_i\} \in E$, we define a demand $D(r, s_i) = (m-1)B - (m-1) * s(a_i)$.

We claim that after N being reconfigured, there is no directed link $e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$ that has load higher than $(m-1)B$ if and only if there exists a valid 3-Partition for the set A. Note that, in our setting, only one direction of each bidirected link needs to carry flow.

Assume A has a valid 3-Partition $A_1, \ldots, A_m$. For each $A_i = \{a_j, a_k, a_f\}$, where $1 \leq i \leq m$, we connect $s_j, s_k, s_f \in S$ to the corresponding nodes f^i_j, f^i_k, f^i_f in the subtree T_i, respectively, by adding configured links. Thus, for each r_i, where $1 \leq i \leq m$, the flow size conveyed by the static link (r_i, r) is decreased by B, which is $(m-1)B$. For each static (directed) link (r, s_j), where $1 \leq j \leq 3m$, it has the load value exactly $(m-1)B$.

On the other hand, we assume that a set of configured links M exists such that no static link, e.g., (r, s_i) for $1 \leq i \leq m$, can have a load more than $(m-1)B$. Clearly, each node s_i must be included in a configured link s.t., at least a flow of size $s(a_i)$ arrives at s_i via a configured link, otherwise (r, s_i) is overflowed. Each configured (directed) link (f^k_i, s_i), where $1 \leq i \leq 3m$ and $1 \leq k \leq m$ can only carry flow for the demand $D(f^k_i, s_i)$; otherwise there must be some flows using static links (r, s_i) to arrive at destination $s_q \in S$, where $i \neq q$ and $1 \leq q \leq 3m$; this causes the load on (r, s_i) more than $(m-1)B$. Thus, a similar argument like the segregated model can be given, and it implies a valid 3-Partition $A_1, \ldots, A_m$. □

Now, it remains to cover intractability for the fourth and remaining routing model:

Theorem 3.6. *The SN-load-optimization reconfiguration problem is strongly NP-hard when the given hybrid network N, before reconfiguration, is a tree of height $h \geq 2$.*

Proof. Given an instance of 3-Partition (A, B, s), we construct an instance of the SN-load-optimization reconfiguration problem as follows: the constructed tree T has the node r as its root, and r has a directed child node r_0. For each element $a_i \in A$, r_0 has a direct child $s_i \in S$. The root r also has m subtrees T_i for $1 \leq i \leq m$. For each subtree T_i, its root is the node r_i, which is the direct child of r; and the root r_i has 3 child nodes $Q^i = \{q_1^i, q_2^i, q_3^i\}$. The tree T constitutes the static (bidirected) links E and nodes V of the hybrid network N. To construct the set of all reconfigurable (bidirected) links $\mathcal{E}$, for each $1 \leq i \leq m$, there is a reconfigurable (bidirected) link $\{q_k^i, s_j\} \in \mathcal{E}$, where $k \in \{1, 2, 3\}$ and $s_j \in S$. Without loss of generality, we set $\forall e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} : C(e) = 1$. Regarding demands D, for each $0 \leq i \leq m$, we have $D(r_i, r) = B$, and for each $a_i \in A$, we have $D(s_i, r_0) = B - s(a_i)$. For each $1 \leq i \leq m$, $D(r_i, r_0) = B$ and $D(r_i, q_k^i) = B/2 + \epsilon$, where $k \in \{1, 2, 3\}$ and $\epsilon < B/2 - \max\{a : a \in A\}$. Clearly, the constructed tree only has a height of two. We claim that after N being reconfigured, there is no (directed) link that has the load more than B if and only if there exists a valid 3-Partition for the set A.

We note that, by our setting, each bidirected link in $E \cup \mathcal{E}$ only needs to carry flow in one direction.

If A has a valid 3-Partition $A_1, \ldots, A_m$, for each $A_i = \{a_j, a_k, a_f\}$, where $1 \leq i \leq m$, we connect $s_j, s_k, s_f \in S$ to the corresponding nodes q_1^i, q_2^i, q_3^i in the subtree T_i, respectively, by adding three configured (bidirected) links, and then we send three flows of sizes $s(a_j)$, $s(a_k)$, and $s(a_f)$ from r_i to r_0 through the configured (directed) links (q_1^i, s_j), (q_2^i, a_k), and (q_3^i, s_f), respectively. For other demands, we send them on their own static links, respectively. Clearly, all demands are served but no link in $\overrightarrow{E} \cup \overrightarrow{M}$ has a load more than B.

Conversely, assume that we have an optimal reconfiguration $M \subseteq \mathcal{E}$ for the SN-reconfiguration problem and a load-optimization flow f for $N(M)$. Without loss of generality, for each $D(r_i, r)$, where $i \in \{0, \ldots, m\}$, we assume that the corresponding flow is only sent on the static (directed) link (r_i, r) in f. If not, some flows for $D(r_i, r) = B$ must also go through (r_j, r), where $j \in \{0, \ldots, m\}$ and $r_j \neq r_i$. Since $D(r_j, r) = B$, to make $L(f(r_j, r)) \leq B$, we know flows serving $D(r_j, r) = B$ must go through (r_i, r) too. Therefore, we can cancel the alternative path for each $D(r_i, r)$, where $i \in \{0, \ldots, m\}$, to force each demand $D(r_i, r)$ only being sent on (r_i, r) without increasing the load value of any (directed) link. For each subtree T_i, where $i \in \{1, \ldots, m\}$, we know flows serving $D(r_i, q_j^i)$, where $j \in \{1, 2, 3\}$, must be only sent on (r_i, q_j^i) due to our assumption. Thus, for each T_i, there must be three configured (directed) links from its three leaf nodes to three nodes in S, otherwise, one static (directed) link (r_i, q_j^i), where $j \in \{1, 2, 3\}$, must overflow after serving $D(r_i, r_0) = B$. To serve each $D(s_i, r_0) = B - s(a_i)$, where $s_i \in S$ and $a_i \in A$, the link (s_i, r_0) already carries a flow of size $B - s(a_i)$, and then each (s_i, r_0) can only convey a flow of size $s(a_i)$ for some demands $D(r_j, r_0)$, where $j \in \{1, \ldots, m\}$. Therefore, for each subtree T_i, we need to make three configured links to generate three (directed) paths: (r_i, q_1^i, s_j, r_0), (r_i, q_2^i, s_k, r_0) and (r_i, q_3^i, s_f, r_0), where $s_k, s_j, s_f \in S$, to convey B flows for $D(r_i, r_0)$, which implies $s(a_j) + s(a_k) + s(a_f) = B$. Finally, each subtree T_i has three configured (bidirected) links connecting three nodes in S, which indicates a valid 3-Partition $A_1, \ldots, A_m$. □

3.3 Non-submodularity

The submodularity of objective functions plays an important role in approximating optimization problems [78], as by Venkatakrishnan et al. [79] for hybrid switch networks. However, their objective function does not consider load-balancing and hence does not apply in our setting, as we show next.

Definition of submodularity. We recall the definition of submodularity [38]: A function $f : 2^B \mapsto \mathbb{R}$, where 2^B is a power set of a finite set B, is submodular if it satisfies that for every

$X, Y \subseteq B$ with $X \subseteq Y$ and every $x \in B \setminus Y$,

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y) .$$

Overview. In this subsection, we investigate the submodularity of the objective function Φ of a τ-reconfiguration problem, which minimizes the maximum load of reconfigured networks $N(M)$, i.e., $L_{\text{min-max}}(N(M))$, for all valid reconfigurations M of a given hybrid network N. Moreover, we are also interested in the submodularity of the objective function Ω that maximizes the gap of the minimized maximum load between the given hybrid network N before reconfiguration and reconfigured networks $N(M)$ for reconfigurations M of N. We will show that both functions Φ and Ω are not submodular functions. To prove the functions to be not submodular, we present special instances as counter-examples.

Theorem 3.7. *For τ-load-optimization reconfiguration problems, where $\tau \in \{US, UN, SS, SN\}$, the objective function Φ that minimizes $L_{min\text{-}max}(N(M))$ for all reconfigurations M of N is not submodular.*

Proof. For a hybrid network $N = (V, E, \mathcal{E}, C)$, we define the set of nodes $V = U \cup Q$, where $U = \{u_i : i = 1, 2, 3\}$ and $Q = \{q_i : i = 1, 2, 3\}$. For static (bidirected) links E, we have six options: $\{u_1, u_2\} \in E$, $\{u_1, u_3\} \in E$, $\{q_1, q_2\} \in E$, $\{q_1, q_3\} \in E$, and $\{u_2, q_2\} \in E$, $\{u_3, q_3\} \in E$. For each $i \in \{1, 2, 3\}$, there is a reconfigurable (bidirected) link $\{u_i, q_i\} \in \mathcal{E}$. W.l.o.g., each (directed) link in $\overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$ has the same capacity $\gamma \in \mathbb{R}^+$. Let our objective function $\Phi : 2^{\mathcal{E}} \mapsto \mathbb{R}^+$ be a function defined by an equation $\Phi(M) = L_{\text{min-max}}(N(M))$, where $M \in 2^{\mathcal{E}}$ is a reconfiguration (matching). Recall the definition of submodularity. We define $X = \{\{u_2, q_2\}\} \subseteq \mathcal{E}$, $Y = \{\{u_1, q_1\}, \{u_2, q_2\}\} \subseteq \mathcal{E}$ and $x = \{u_3, q_3\} \in \mathcal{E}$. When $\tau =$ SS, SN, we define demands as follows: $D(u_3, q_3) = 3$, $D(u_2, q_2) = 3$, $D(q_2, u_2) = 3$ and $D(u_1, q_1) = 3$. When $\tau =$ SS, we have $\Phi(X \cup \{x\}) = \frac{9}{4\gamma}$, $\Phi(X) = \frac{3}{\gamma}$, $\Phi(Y \cup \{x\}) > \frac{3}{2\gamma}$, and $\Phi(Y) \leq \frac{9}{4\gamma}$. Thus, the Inequality (1) shows that the function Φ is not submodular.

$$\Phi(X \cup \{x\}) - \Phi(X) = \frac{-3}{4\gamma} < \Phi(Y \cup \{x\}) - \Phi(Y) > \frac{-3}{4\gamma} . \tag{1}$$

When $\tau =$ SN, we have $\Phi(X \cup \{x\}) = \frac{9}{4\gamma}$, $\Phi(X) = \frac{3}{\gamma}$, $\Phi(Y \cup \{x\}) = \frac{9}{5\gamma}$, and $\Phi(Y) = \frac{9}{4\gamma}$. Thus, the Inequality (2) shows that the function Φ is not submodular.

$$\Phi(X \cup \{x\}) - \Phi(X) = \frac{-3}{4\gamma} < \Phi(Y \cup \{x\}) - \Phi(Y) = \frac{-1.8}{4\gamma} . \tag{2}$$

When $\tau =$ US, we modify our above constructed network N by adding one more node d and one more static link $\{d, q_2\} \in E$, while reconfigurable links $\mathcal{E}$ are unchanged. We define new demands as follows: $D(u_3, q_3) = 3$, $D(u_2, d) = 3$, and $D(u_1, q_1) = 3$. Now, we have $L_{\text{min-max}}(N(\emptyset)) = 6$. When $\tau =$ US, we know $\Phi(X \cup \{x\}) = 3$, $\Phi(X) = 6$, $\Phi(Y \cup \{x\}) = 3$, and $\Phi(Y) = 3$. Thus, the Inequality (3) shows that the function Φ is not submodular.

$$\Phi(X \cup \{x\}) - \Phi(X) = -3/\gamma < \Phi(Y \cup \{x\}) - \Phi(Y) = 0 . \tag{3}$$

When $\tau =$ UN, we extend the above constructed network N for US by adding one more static link $\{d, q_1\} \in E$ and removing the static link $\{q_1, q_2\} \in E$, while reconfigurable links $\mathcal{E}$ are unchanged. We define new demands as follows: $D(u_3, q_3) = 3$, $D(u_2, d) = 3$, and $D(u_1, d) = 3$. Now, we know $\Phi(X \cup \{x\}) = 3/\gamma$, $\Phi(X) = 6/\gamma$, $\Phi(Y \cup \{x\}) = 3/\gamma$, and $\Phi(Y) = 3/\gamma$. Thus, the function Φ is not submodular. □

Theorem 3.8. *For τ-load-optimization reconfiguration problems, where $\tau \in \{US, UN, SS, SN\}$, the objective function Ω that maximizes $L_{min\text{-}max}(N(\emptyset)) - L_{min\text{-}max}(N(M))$ for all reconfigurations M of N, is not submodular.*

Proof. For a hybrid network $N = (V, E, \mathcal{E}, C)$, we define nodes $V = U \cup Q \cup P$, where $U = \{u_i : i = 1, 2, 3\}$, $P = \{p_i : i = 1, 2, 3\}$ $Q = \{q_i : i = 1, 2, 3\}$; For each $i \in \{1, 2, 3\}$, we set two static links $\{u_i, q_i\} \in E$ and $\{p_i, u_i\} \in E$, and a reconfigurable link $\{u_i, q_i\} \in \mathcal{E}$, and two demands $D(u_i, q_i) = 3$ and $D(p_i, q_i) = 3$. Let our objective function $\Omega : 2^{\mathcal{E}} \mapsto \mathbb{R}^+$ be defined by an equation $\Omega(M) = \omega - L_{\text{min-max}}(N(M))$, where a reconfiguration (matching) $M \in 2^{\mathcal{E}}$ and $\omega = L_{\text{min-max}}(N(\emptyset))$. With the loss of generality, for each (directed) link in $\overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$, it has the capacity γ. Recall the definition of submodularity. We set $X = \{\{u_1, q_1\}\}$, $Y = \{\{u_1, q_1\}, \{u_2, q_2\}\}$ and $x = \{u_3, q_3\}$. For each routing model $\tau \in \{\text{US}, \text{UN}, \text{SS}, \text{SN}\}$, we always have $\Omega(X \cup \{x\}) = \omega - 6/\gamma$, $\Omega(X) = \omega - 6/\gamma$, $\Omega(Y \cup \{x\}) = \omega - 3/\gamma$, and $\Omega(Y) = \omega - 6/\gamma$. Hence, Ω is not submodular. □

4 HYBRID SWITCH NETWORKS

As we saw before, already tree networks of height ≥ 2 are NP-hard to optimize, and optimizations leveraging submodularity are not possible. Yet it is worth noting that the NP-hardness for stars, i.e., trees of depth one, is still open since the NP-hardness established for trees of height ≥ 2 collapses on simple structures of star topologies. In fact, many NP-hard problems can become tractable after restricting the input graphs, e.g., the minimum vertex cover becomes polynomially solvable on trees by using dynamic programming [19]. This raises the interesting question if we can obtain *optimal* and *polynomial-time* algorithms for a data center network that can be abstracted as a star topology.

4.1 Non-blocking Data Center Topologies

Common data center topologies have trees of height 2 as subgraphs or minors and hence seem like bad candidates for efficient algorithms at first glance. However, already early designs adapted from telecommunications such as Clos [18] topologies have a so-called non-blocking property, which we can use to our advantage. An interconnecting topology C is non-blocking, if the servability of a flow from v_1 to v_2 via C only depends on the utilization of the links (v_1, C) and (C, v_2) : "**such an interconnect behaves like a crossbar switch**" [89]. In other words, from a load-utilization perspective, the maximum load inside C will *not* be higher than on the egress/ingress links of C. Non-blocking interconnects have hence become popular data center topologies [4] in particular in the form of folded Clos networks or Fat-Trees [54], depicted in Figure 2(a): the actual topology inside the interconnect (marked in a blue rectangle) is immaterial and we only need to consider the links incident to the nodes[4]—a fact commonly used, e.g., for bandwidth guarantees of the hose model [25] in Clos topologies [40, Section 4.1].

Thus, for our purposes, we can abstract the data center interconnect C (which can be understood as a packet switch) by a single center node c, leaving our previous intractability considerations behind. We hence turn our attention to *hybrid switch networks* as considered by of Venkatakrishnan et al. [79], which are represented by a packet and a circuit switch connected to all nodes, see Figure 2(b).

Routing in hybrid switch networks is straightforward (only one path exists for each node pair in the packet switched network), but the addition of a circuit switch adds a large degree of freedom: First, the number of possible matchings grows exponentially, and second, we have to decide for each flow which path to take as well. Notwithstanding, the special structure of hybrid switch networks allows us to solve reconfiguration and routing efficiently.

[4]We note that the non-blocking property can also be restricted to keep distributed routing schemes in mind, we refer to Yuan [89] for an in-depth discussion.

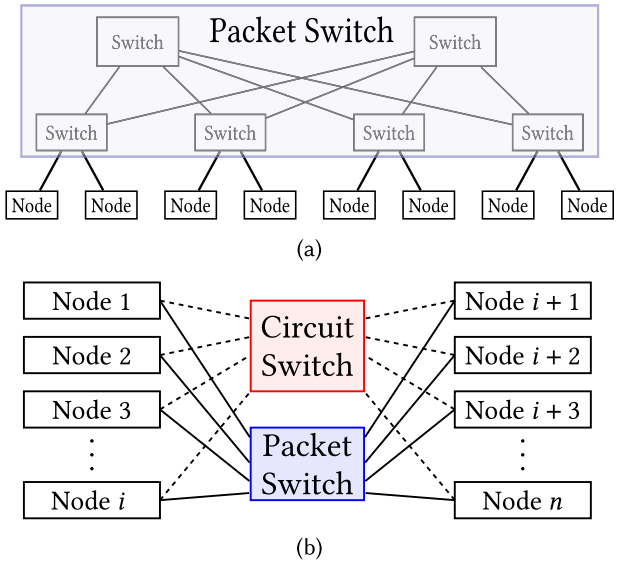


Fig. 2. Illustration of a Fat-Tree network in 2(a) and a hybrid switch network in 2(b), as in [79, Figure 1]. Due to the non-blocking property of network in 2(a), we can abstract the interconnect enclosed by the blue box as a packet switch, as depicted in 2(b). Additionally, a hybrid switch network also contains a reconfigurable circuit switch (e.g., an OCS) that provides a matching of the nodes, to be optimized for the demands D. Hence, the augmentation of a Fat-Tree by an OCS can also be investigated from the viewpoint of hybrid switch networks.

(a) Classic Matching Setting (b) Red-Target Matching Setting

Fig. 3. In 3(a), we can find a maximal matching M by setting either $M = \{\{a, c\}\}$ or $M = \{\{a, e\}\}$. However, in 3(b), we need to find a matching such that all red nodes are included, and hence we cannot find any RTM to cover the red nodes c and e: as a may only be part of one matching edge, we only cover either c or e, but not both.

We structure our approach as follows. We first introduce an auxiliary problem in Section 4.2 and a constant-time triangle graph algorithm in Section 4.3, which we then leverage for our optimal algorithm in Section 4.4. We lastly discuss performance bounds and extensions in Section 4.5.

4.2 Red-target Matching

As each configuration of an OCS must be a matching, we cannot simultaneously create a reconfigurable connection for each demand. Still, intuitively, it is desirable to relieve the nodes, respectively, node-pairs with higher communication intensities by reconfigurable links. Later in our algorithms, we will mark some nodes (in red) which must be connected to the OCS in order to satisfy a given load threshold. However, not all reconfigurations, i.e., matchings, are suitable for such a task. Given such red-colored nodes, the question is if all such red nodes can be matched accordingly, which is formalized in Definition 4.1:

*Definition 4.1 (**Red-Target Matching (RTM**)).* Given a graph $G = (V, E)$, where a subset of nodes $V' \subset V$ are colored as "red", the question is to find a matching M of G such that each colored node $v \in V'$ is covered by an edge of M.

To illustrate Definition 4.1, we introduce an example shown in Figure 3. The RTM problem looks for a restricted matching, which not only satisfies the degree-bound of a matching but also contains the set of all colored nodes $V' \subseteq V$.

Lemma 4.2. *The RTM problem (Definition 4.1) can be solved by a maximum-weight matching algorithm in polynomial time.*

Proof. For a given graph $G = (V, E)$, if an edge $e \in E$ does not contain a red node in its endpoints, then it can be removed directly. For each edge $e \in E$ having both endpoints of the color red, we set the weight $w(e) = 2$, and for each edge $e \in E$ that has only one endpoint of the color red, we set $w(e) = 1$. Let the number of red-colored nodes V' in G be n. If we can find a matching M of the weight n, then all of these n red nodes are contained in M. There is no matching that can have a weight more than n, otherwise the number of red-colored nodes V' in G is more than n. Therefore, if the RTM problem has a solution then a maximum-weight algorithm can always find a valid matching M for RTM. If the maximum-weight matching has a weight of less than n, then RTM has no solution. Lastly, a maximum-weight matching is solvable in polynomial time, e.g., by the Blossom algorithm [26], which has a running time $O(|E||V|^2)$. □

4.3 Selection of Suitable Reconfigurable Links

In the studied hybrid switch networks, reconfigurable links can be created between any pair of nodes connected to the packet switch, e.g., via an OCS. While we will select the (matching) subset of reconfigurable (bidirected) links in the next subsection, we herein identify the benefit of adding specific reconfigurable links.

Lemma 4.3. *Given a reconfigured network $N(M)$, which is a triangle on three nodes $V = \{a, b, c\}$ with the only configured (bidirected) link $\{a, b\} \in \mathcal{E}$ and two static (bidirected) links $\{c, a\}, \{c, b\} \in E$, then for demands D, a load-optimization flow in $N(M)$ can be computed in a constant time under routing models $\tau \in \{US, SS, SN\}$.*

Proof. In the triangle $N(M)$, there are at most six demands in D and six directed links $\overrightarrow{E} \cup \overrightarrow{\mathcal{E}}$. For each demand, e.g., $D(a, b)$, the directed link (a, b) is called the shortcut of $D(a, b)$, and the directed path (a, c, b) from a to b is called the indirect path of $D(a, b)$.

For the segregated routing model, demands $D(c, a)$, $D(a, c)$, $D(c, b)$, and $D(b, c)$ can be only sent on their shortcuts (c, a) (a, c), (c, b), and (b, c), respectively, which are static links in $\overrightarrow{E}$, and then we only need to consider $D(a, b)$ and $D(b, a)$. Moreover, for the unsplittale routing model, each demand, e.g., $D(a, b)$, can only be sent on a single path: either its shortcut (a, b) or its indirect path (a, c, b).

When $\tau = \text{US}$, $D(a, b)$ can only be sent through either (a, b) or (a, c, b), and the similar argument can be applied to $D(b, a)$. In terms of the given capacity function C, a load-optimization flow can be decided by searching these four different routing possibilities for $D(a, b)$ and $D(b, a)$, which is in a constant time.

When $\tau = \text{SS}$, the respective load values for directed links (a, c), (c, b), and (a, b) only depends on how the demand $D(a, b)$ divides its traffic between the indirect path (a, c, b) and its shortcut (a, b), while an analogous argument can be applied to the demand $D(b, a)$. Thus, a load-optimization flow can be decided in a constant time.

When $\tau = \text{SN}$, a load-optimization flow in the triangle $N(M)$ can be computed in constant time as well. The details are given in Lemma A.1, deferred to the Appendix in order to improve readability. □

It remains to utilize the single triangle algorithms in a larger context: Lemma 4.4 shows that the optimal flow computed locally in each triangle $\{v_i, c, v_j\}$ provides a lower bound for the subflow

ALGORITHM 1: Preprocess Triangles

Input : Hybrid switch network $N = (V, E, \mathcal{E}, C)$, demands D and a routing model $\tau \in \{\text{US}, \text{SS}, \text{SN}\}$;
Output: A set $S^\Delta := \{(\mu^\Delta_{ij}, f^\Delta_{ij}) : \{v_i, v_j\} \in \mathcal{E}\}$, s.t., $\forall(\mu^\Delta_{ij}, f^\Delta_{ij}) \in S^\Delta$ has $\mu^\Delta_{ij} \le L_{\text{min-max}}(N(M))$ for an optimal reconfiguration $M \subseteq \mathcal{E}$ and D;

$S^\Delta := \emptyset$;
for *each reconfigurable (bidirected) link* $\{v_i, v_j\} \in \mathcal{E}$, *where* $v_i, v_j \in V$ *are leaf nodes in* N **do**
  define the triangle on nodes $\{v_i, v_j, c\}$, where c is the center node, and the new demands D' on $\{v_i, v_j, c\}$;
  let $D'(v_i, v_j) = D(v_i, v_j)$ and $D'(v_j, v_i) = D(v_j, v_i)$;
  for *each node* $u \in \{v_i, v_j\}$ **do**
    let $D'(u, c) = \sum_{v \in V \setminus \{v_i, v_j\}} D(u, v)$;
    let $D'(c, u) = \sum_{v \in V \setminus \{v_i, v_j\}} D(v, u)$;
  for $\tau \in \{\text{US}, \text{SS}, \text{SN}\}$, by Lemma 4.3, a load-optimization flow f^Δ_{ij} in the triangle $\{v_i, v_j, c\}$ to serve demands D' can be computed in constant time;
  $S^\Delta = S^\Delta \cup \{(\mu^\Delta_{ij}, f^\Delta_{ij})\}$, where $\mu^\Delta_{ij} := L_{max}(f^\Delta_{ij})$;
return S^Δ;

of a globally optimal flow of the hybrid switch network N and demands D in the same triangle, and Lemma 4.5 further indicates that a globally optimal flow can be obtained by combining these locally optimal flows in triangles.

Lemma 4.4. *Given a hybrid switch network N on leaves $V' = \{v_1, \ldots, v_n\}$, where we denote the central packet switch by a node c, demands D, and a routing model $\tau \in \{US, SS, SN\}$, for any reconfiguration M' of N, let f be* an arbitrary flow *serving D in the reconfigured network $N(M')$. Let $\{v_i, v_j\} \in M'$ be any configured (bidirected) link in M', where $v_i, v_j \in V'$. For the triangle on $\{v_i, v_j, c\}$, let E^Δ_{ij} denote the six (directed) links of this triangle, i.e., $E^\Delta_{ij} \subset \overrightarrow{E} \cup \overrightarrow{M}$, and let μ^Δ_{ij} be the minimized maximum load computed by Algorithm 1 for the triangle $\{v_i, v_j, c\}$. We then obtain:*

$$\max\left\{L\left(f\left(e\right)\right) : e \in E^\Delta_{ij}\right\} \ge \mu^\Delta_{ij}\,. \tag{4}$$

Proof. Let the given hybrid switch network N have nodes $V = V' \cup \{c\}$, where c is the central packet switch node and $V' = \{v_1, \ldots, v_n\}$ are leaf nodes (leaves). Recall that a reconfiguration M' must be a matching. Thus, in the reconfigured network $N(M')$, for each configured (bidirected) link $\{v_i, v_j\} \in M'$, where $v_i, v_j \in V'$, the node v_i (respectively, v_j) only connects to nodes c and v_j (respectively, v_i). Let f be an arbitrary flow serving D in $N(M')$. Any partial flow of f that start from the node $u \in \{v_i, v_j\}$ and end at a node $v \in V \setminus \{v_i, v_j\}$ must go through the center c to leave the triangle $\{v_i, c, v_j\}$, and the size of these flows must be $D'(u, c)$ defined in Algorithm 1; on the other hand, any partial flow (sub-flow) of f that starts from a node $v \in V \setminus \{v_i, v_j\}$ and ends at the node $u \in \{v_i, v_j\}$ must go through the center c to enter the triangle $\{v_i, c, v_j\}$ and the size of these sub-flows must be $D'(c, u)$ defined in Algorithm 1. Therefore, the local sub-flows of f inside the triangle $\{v_i, v_j, c\}$ satisfy the demands D' defined in Algorithm 1. Since μ^Δ_{ij} denotes the maximum load of a local load-optimization flow serving D' in $\{v_i, v_j, c\}$, then by the correctness of Lemma 4.3, In Equation (4) holds directly. □

Lemma 4.5. *In Algorithm 2, a load-optimization flow f_{final} serving D in a reconfigured network $N(M)$ under a routing model $\tau \in \{US, SS, SN\}$ can be constructed in a runtime of $O(n^2)$, where the number of demands D is at most n^2.*

ALGORITHM 2: Reconfiguration for Hybrid Switch Networks

Input : A hybrid switch network $N = (V, E, \mathcal{E}, C)$, with nodes $V = \{v_1, \ldots, v_n, c\}$, where c is the center node having leaves $\{v_i, \ldots, v_n\}$, static links $E = \{\{v_i, c\} : v_i \in V \setminus \{c\}\}$, demands D, and a routing model $\tau \in \{\text{US}, \text{SS}, \text{SN}\}$;

Output : A reconfiguration $M \subseteq \mathcal{E}$ and a load-optimization flow f_{final} of $N(M)$ for D;

find an original (optimal) flow $f_{\text{old}} : \overrightarrow{E} \mapsto \mathbb{R}^+$ serving D on N before reconfiguration;
run Algorithm 1 on (N, D, τ) to obtain S^{Δ};
run Algorithm 3 on the input $(N, D, f_{\text{old}}, S^{\Delta})$;
if *Algorithm 3 returns "null"* **then**
    return "null";
else
    let $M \subseteq \mathcal{E}$ be a reconfiguration returned by Algorithm 3;
    construct a load-optimization flow f_{final} for $N(M)$, D and τ, based on S^{Δ} (Lemma 4.5);
    return a reconfiguration M and f_{final};

Proof. We note that a local load-optimization flow f_{ij}^{Δ} for each triangle $\{v_i, v_j, c\}$ where $\{v_i, v_j\} \in \mathcal{E}$, has been computed in Algorithm 1 according to τ and D'.

Let M be an optimal reconfiguration for the hybrid switch network N and demands D. For each configured (bidirected) link $\{v_i, v_j\} \in M$, the set S^{Δ} returned by Algorithm 1 contains a load-optimization flow f_{ij}^{Δ} for the triangle $\{v_i, v_j, c\}$ and D'. We first construct the related sub-flow serving an arbitrary demand $D(v_i, v_j)$ in f_{final}, where v_i and v_j are leaf nodes. First, if $\{v_i, v_j\} \in M$, then the flow for $D(v_i, v_j)$ (respectively, $D(v_j, v_i)$) is already given in f_{ij}^{Δ} contained in S^{Δ}. Second, if there are two configured links $\{v_i, v_k\}$ and $\{v_j, v_l\}$ in M, then the flow of $D(v_i, v_j)$ obtained by merging the sub-flow of size $D(v_i, v_j)$ in f_{ik}^{Δ} serving $D'(v_i, c)$ and the sub-flow of size $D(v_i, v_j)$ in f_{jl}^{Δ} serving $D'(c, v_i)$ on the joint center c. If only v_i is contained in a configured link $\{v_i, v_l\} \in M$, then the sub-flow serving $D(v_i, v_j)$ in f_{final} is obtained by extending the sub-flow of size $D(v_i, v_j)$ in f_{il}^{Δ} serving $D'(v_i, c)$ from the destination c to the node v_j. Moreover, for the demand $D(v_i, c)$ (respectively, $D(c, v_i)$), if v_i is contained in a configured link $\{v_i, v_k\} \in M$, then the sub-flow serving $D(v_i, c)$ (respectively, $D(c, v_i)$) in f_{final} can be found in the local flow f_{ik}^{Δ} in S^{Δ} directly; otherwise, we send its flow directly on the static (directed) link (v_i, c) (respectively, (c, v_i)) in f_{final}.

Lastly, for each demand, its flow can be constructed in constant time in Algorithm 2. Thus, the running time to construct f_{final} relies on the number of demands D, which is $O(n^2)$. □

4.4 Solving Hybrid Switch Networks Optimally

We now combine our previous results to optimally solve the reconfiguration problem on hybrid switch networks.[5]

Theorem 4.6. *If each reconfigurable link in $\mathcal{E}$ is only between two leaf nodes, then the τ-reconfiguration problem on hybrid switch networks is solved optimally by Algorithm 2 when $\tau \in \{US, SS, SN\}$.*

Proof. For a hybrid switch network N, let M be an optimal reconfiguration for N, and let $\mu_{\min} = L_{\text{min-max}}(N(M))$ denote the minimized maximum load of the reconfigured network $N(M)$. Note that such an optimal reconfiguration M always exists for hybrid switch networks N, when N has at least two leaf nodes. Thus, we prove that Algorithm 2 can find such an optimal solution M.

[5]Recall that the UN model is NP-hard on hybrid switch networks (Section 3.2).

ALGORITHM 3: Determine Reconfiguration

Input : A hybrid switch network N, demands D, the original flow f_{old}, and a set S^{Δ};
Output: A RTM matching (reconfiguration) $M \subseteq \mathcal{E}$ or "null";

let $V' := V \setminus \{c\}$ be all leaf nodes in N;
let $T := \max\{\mu_{ij}^{\Delta} : (\mu_{ij}^{\Delta}, f_{ij}^{\Delta}) \in S^{\Delta}\} \bigcup \{L(f_{\text{old}}(e)) : e \in \overrightarrow{E}\}$ be sorted in an ascending order;
for *each value* $\mu \in T$ **do**
    create an extra graph (clique) G on leaf nodes V', s.t., $E(G) = \{\{v_i, v_j\} : v_i, v_j \in V'\}$;
    for *each leaf node* $v_i \in V'$ **do**
        if *the load* $L(f_{\text{old}}(v_i, c)) > \mu \vee L(f_{\text{old}}(c, v_i)) > \mu$ **then**
            color the corresponding node $v_i \in V'$ by "red";
    if *a RTM matching (Def. 4.1)* M *can be computed on* G *to cover all "red" nodes* **then**
        return M;
return "null";

For a load-optimization flow f_{opt} of $N(M)$, there must be at least a directed link e^* in $N(M)$ such that $L(f_{\text{opt}}(e^*)) = \mu_{\min}$. First, if there is e^* contained in a triangle $\{v_{i^*}, c, v_{j^*}\}$ in $N(M)$, where $v_{i^*}, v_{j^*} \in V$ are leaf nodes, then it implies that $\mu_{\min} = \mu_{i^*j^*}^{\Delta}$ by Lemma 4.3-4.4, where $\mu_{i^*j^*}^{\Delta} = L_{\max}(f_{i^*j^*}^{\Delta})$ and f_{ij}^{Δ} denotes a load-optimization flow f_{ij}^{Δ} serving D' computed by Algorithm 1. We can further imply that $\mu_{\min} = \mu_{i^*j^*}^{\Delta} = \max\{\mu_{ij}^{\Delta} : (\mu_{ij}^{\Delta}, f_{ij}^{\Delta}) \in S^{\Delta}\}$, otherwise there must be another triangle $\{v_{i'}, c, v_{j'}\}$ that has $\mu_{i'j'}^{\Delta} > \mu_{\min}$, contradicting that $\mu_{\min}$ is the maximum load on f_{opt}. On the other hand, if e^* is a static link and no configured link in M is incident with e^* on a leaf node, then $\mu_{\min}$ must be already in $\{L(f_{\text{old}}(e)) : e \in \overrightarrow{E}\}$, which also has $\mu_{\min} \geq \max\{\mu_{ij}^{\Delta} : (\mu_{ij}^{\Delta}, f_{ij}^{\Delta}) \in S^{\Delta}\}$. Thus, $\mu_{\min}$ must be included in T in Algorithm 3. Since the binary search goes through T exhaustively, then $\mu_{\min}$ can be always detected and used as a threshold for Algorithm 3 to search for a matching.

Now, we prove that, when each reconfigurable link in $\mathcal{E}$ is between two leaf nodes in V, given a threshold $\mu_{\min}$, Algorithm 2 can find an optimal reconfiguration M for a hybrid switch network N and a flow f serving D in $N(M)$ such that $L_{\max}(f) \leq \mu_{\min}$.

Before reconfiguration, on the original flow f_{old}, for each static (bidirected) link $\{v_i, c\} \in E$, where $v_i \in V \setminus \{c\}$, if it has $L(f(v_i, c)) > \mu_{min}$ or $L(f(c, v_i)) > \mu_{min}$, then its leaf node v_i must be contained in a configured link in M, otherwise, the loads on (v_i, c) and (c, v_i) are unchanged after reconfiguration. Thus, in Algorithm 3, we color such nodes by "red" and try to find a matching to cover all "red" nodes. Lemma 4.1 ensures that a matching M covering all "red" nodes in G must be detected if it exists. Due to the way of constructing G, for each $\{v_i, v_j\} \in M$, Lemma 4.3 implies that the local load-optimization flow f_{ij}^{Δ} in $\{v_i, v_j, c\}$ serving D' has the maximum load $L_{\max}(f_{ij}^{\Delta}) \leq \max\{\mu_{ij}^{\Delta} : (\mu_{ij}^{\Delta}, f_{ij}^{\Delta}) \in S^{\Delta}\} \leq \mu_{\min}$. Lemma 4.5 guarantees that a load-optimization flow f_{final} serving D in the $N(M)$ can be constructed such that $L_{\max}(f_{\text{final}}) \leq \mu_{\min}$. Note that for any static link $\{v_k, c\} \in E$, if v_k is not contained in any configured link of M, then $L(f_{\text{final}}(v_k, c)) = L(f_{\text{old}}(v_k, c)) \leq \mu_{\min}$ and $L(f_{\text{final}}(c, v_k)) = L(f_{\text{old}}(c, v_k)) \leq \mu_{\min}$. □

We now briefly show that our algorithms also extend to the case where we can create a reconfigurable link to the central packet switch and also bound the runtime:

THEOREM 4.7. *The τ-reconfiguration problem on hybrid switch networks is solved optimally by Algorithm 2 in a polynomial time $O(\beta \cdot \log n)$, where n is the number of nodes and β denotes the running time of a maximum matching algorithm, when $\tau \in \{US, SS, SN\}$.*

Proof. Theorem 4.6 has shown the correctness under the restriction, where each reconfigurable link in $\mathcal{E}$ must be between two leaf nodes. Now, we will show that a τ-reconfiguration problem can still be solved by Algorithm 2 without the restriction.

If $\mathcal{E}$ contains a reconfigurable link $\{v_i, c\}$, where v_i is a leaf node and c is the center, we could create an additional leaf node v_{n+1} in V. To introduce additional demands, for each $u \in V \setminus \{v_{n+1}\}$, we define $D(v_{n+1}, u) := 0$ and $D(u, v_{n+1}) := 0$. Then for each reconfigurable link $\{v_i, c\} \in \mathcal{E}$, we remove it from $\mathcal{E}$ and add a new reconfigurable link $\{v_i, v_{n+1}\}$ into $\mathcal{E}$. For the matching M and a load-optimization flow f_{final} returned by Algorithm 2, if $\{v_i, v_{n+1}\} \in M$, then remove it and add $\{v_i, c\}$ into M, and update f_{final} by moving flow on the directed path (v_i, v_{n+1}, c) (respectively, (c, v_{n+1}, v_i)) to the existing configured link (v_i, c) (respectively, (c, v_i)). If no link in $\mathcal{E}$ contains the center c, then Theorem 4.6 has proved the correctness.

Now, we assume that there must be at least one reconfigurable link in $\mathcal{E}$ containing c. Note that the original $\mu_{\min}$ can be still stored in T after the above processing. Thus, the τ-reconfiguration problem on hybrid switch networks is solved optimally by Algorithm 2.

Runtime analysis. Algorithm 1 computes local demands D' and the corresponding optimal flow locally in each triangle $\{v_i, c, v_j\}$ of $\{v_i, v_j\} \in \mathcal{E}$, leading to the runtime of $O(|V| \cdot |\mathcal{E}|)$. The analysis of Algorithm 3 reveals that its runtime is dominated by (1) the binary search over values of T, and (2) finding a maximum matching for each value of T. Hence, given the overhead of binary search $O(\log |T|) \in O(\log n)$, where $|T| = |V| + 1$, the time cost of Algorithm 3 is $O(\beta \cdot \log |T|)$, where β denotes the time of computing a maximum matching, e.g., $\beta = O(|\mathcal{E}||V|^2)$ by Blossom algorithm [26]. Finally, by using Algorithms 1 and 3 as subroutines, Algorithm 2 has its runtime bounded by $O(|V| \cdot |\mathcal{E}| + \beta \cdot \log |T|)$, dominated by $O(\beta \cdot \log n)$. □

For example the original Blossom algorithm [26] can be used compute a maximum weight matching in $\beta = O(|E||V|^2)$, but faster maximum weight matching algorithms exist, for which we refer to the comprehensive overview by Duan and Pettie [24, Tbl. III].

4.5 Bounds and Extensions

Given that we provided optimal algorithms for hybrid switch networks above, we now investigate theoretical performance bounds and extensions. As such, we provide bounds on the improvement of the load after reconfiguration, prove that maximum matching algorithms do not perform well in terms of competitive analysis, and show how our algorithms can be extended to multiple small reconfigurable switches.

Improvement bounds. If the capacities of reconfigurable links are arbitrarily large, in comparison to the static links, then the maximum load after applying reconfiguration can become arbitrarily small, under selected scenarios. Thus, to understand the intrinsic lower bounds of the reconfiguration problem on hybrid switch networks $N = (V, E, \mathcal{E}, C)$, we investigate the case where the capacity function C is uniform, denoted by $(V, E, \mathcal{E}, 1)$.

For a hybrid network N with uniform capacities, the improvement of the load on an arbitrary static link $\{u, v\} \in E$ relies on the incremental edge-connectivity imposed by the reconfigured links in M between u and v in $N(M)$. If a node u has only one static link $\{u, v\} \in E$, then the edge-connectivity from u to v can be at most two in $N(M)$ for any reconfiguration M, which further implies that the load on edges outgoing from u can at best be split along both edges after performing reconfiguration.

Lemma 4.8. *Given a hybrid switch network $N = (V, E, \mathcal{E}, 1)$, demands D and a routing model $\tau \in \{US, UN, SS, SN\}$, for any reconfiguration M of N, we have $L_{min\text{-}max}(N(M)) \geq L_{min\text{-}max}(N(\emptyset))/2$.*

Proof. Given a hybrid switch network $N = (V, E, \mathcal{E}, 1)$, demands D and a routing model $\tau \in \{\text{US, UN, SS, SN}\}$, let $M^* \subseteq \mathcal{E}$ be an optimal reconfiguration of N and let $f_{\text{opt}}^{M^*}$ be an arbitrary

load-optimization flow for the reconfigured network $N(M^*)$. Let f_{old} be the original flow serving D in N. There must be a leaf node $v_i \in V$ such that a static (directed) link, w.l.o.g., denoted by (v_i, c), has $L(f_{\text{old}}(v_i, c)) = L_{\text{min-max}}(N(\emptyset))$. We know that v_i must be included in a configured link, denoted by $\{v_i, v_j\} \in M^*$, otherwise we still have that $L_{\text{min-max}}(N(M^*)) = L_{\text{min-max}}(N(\emptyset))$ holds. In the triangle $\{v_i, v_j, c\}$, there are at most two link-disjoint directed paths from another node in $\{v_j, c\}$ to the node v_i. We know the size of the flow on the static link (v_i, c) can be at most decreased by half to obtain optimality, which implies $L_{\text{min-max}}(N(M^*)) \geq L(f_{\text{opt}}^{M^*}(v_i, c)) \geq L(f_{\text{old}}(v_i, c))/2$. Thus, for any reconfiguration M of N, we know $L_{\text{min-max}}(N(M)) \geq L_{\text{min-max}}(N(\emptyset))/2$. □

Competitivity of matching algorithms. We next investigate the theoretical performance of a maximum matching algorithm, as e.g., utilized in [82]. The idea based on a maximum matching is that for each reconfigurable link $\{u, v\} \in \mathcal{E}$, we send all flows of demands $D(u, v)$ and $D(v, u)$ on links (u, v) and (v, u) , respectively, then to find a maximum matching to maximize total size of flows on a set of configured links M. As it turns out, such an optimization might yield nearly no benefit, even though an optimal algorithm could hit the theoretical lower bound provided in Lemma 4.8.

Lemma 4.9. *For a τ-load-optimization reconfiguration problem on a hybrid switch network N, where $\tau \in \{US, SS\}$, a maximum matching algorithm can find a reconfiguration M of N, s.t.,*

$$L_{min\text{-}max}(N(\emptyset)) - L_{min\text{-}max}(N(M)) \leq \epsilon, \text{ for an arbitrarily small } \epsilon \geq 0,$$

but where an optimal reconfiguration M^* implies $L_{min\text{-}max}(N(M^*)) = L_{min\text{-}max}(N(\emptyset))/2$.

Proof. Recall the definition of segregated routing. Given a small value $\epsilon \geq 0$, we construct a hybrid switch network $N = (V, E, \mathcal{E}, C)$, where $V = \{v_1, \ldots, v_n, a, b, c, d\}$, c is the center, and other nodes are leaves. For any two nodes $u, v \in V \setminus \{c\}$, we construct a reconfigurable link $\{u, v\} \in \mathcal{E}$. Here, $\forall e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} : C(e) = 1$. Regarding demands D, for each $v_i \in \{v_1, \ldots, v_n\}$, we define $D(v_i, a) = \epsilon$ and $D(b, a) = n\epsilon$, $D(b, d) = n\epsilon$. Clearly, $L_{\text{min-max}}(N(\emptyset)) = 2n\epsilon$. For the maximum matching algorithm, two reconfigurable links $\{b, d\} \in \mathcal{E}$ and $\{v_i, a\} \in \mathcal{E}$, where $i \in \{1, \ldots, n\}$, must be included in a reconfiguration M, which gives $L_{\text{min-max}}(N(M)) = (2n - 1)\epsilon$. However, by selecting $\{a, b\}$ into M^*, we can have $L_{\text{min-max}}(N(M^*)) = n\epsilon = L_{\text{min-max}}(N(\emptyset))/2$. Please note that for the above example, the splittable and unsplittable models show the same results. □

Extension to smaller reconfigurable circuit switches. In case the number of ports of a single reconfigurable switch does not suffice for all nodes in the network, our algorithms also extend to the case of multiple smaller reconfigurable switches. We can connect subsets of the nodes to a reconfigurable switch each, e.g., grouped by historical data w.r.t. the traffic demands. Our hybrid switch algorithms in Section 4 then take this subset of possible reconfigurable links to work with and proceed as usual, e.g., by assigning non-allowed links a weight (benefit) of 0 in matchings.

4.6 Practical Considerations

For non-blocking[6] data-center topologies, where the load-balancing is usually dominated by the last hop, e.g., for incast [69], we can abstract the static topology as a star (tree of depth one) as shown in Figure 2, such that our algorithm can minimize the loads by, e.g., taking away elephant flow from the original static network to high-capacity reconfigurable links. Our solution provides

[6]If the DC topology is blocking, such as, e.g., DCell, Jellyfish, MDCube etc. [53], i.e., in particular in server-centric proposals, we cannot directly apply our algorithms, unless the topologies are augmented to be non-blocking. An extension of our optimal polynomial-time algorithms to general server-centric topologies is unlikely, as we have shown that already simple topologies beyond stars induce intractability.

an efficient and optimal way to design reconfigurable networks for existing DCNs to optimize load-balancing, which significantly outperforms conventional methods of implementing reconfigurable links by a maximum weighted matching, e.g., [29, 82], and by a greedy approach, e.g., [44, 91], as we will show in the next section in practical evaluations, beyond the previous theoretical results.

Our solution can be implemented directly and is generally compatible with pre-installed routing configurations of existing data-centers, as it relies on analyzing the matrix of traffic demands to determine which pairs of intensive-communication nodes to be transferred to reconfigurable links. More specifically, after preprocessing on demands, elephant flows can be separately sent on reconfigurable links, and other remaining demands will be still routed through the static network as before, e.g., ECMP, packet-based routing, flowlet-based routing, and so on can be applied in these settings for the remaining flows.

For real-time applications, the reconfiguration delay (time) that reconfigurable links cannot transfer data during their establishing phase might degenerate the performance when traffic pattern changes very significantly in a short interval. However, in general, data-center traffic patterns feature significant temporal locality,[7] and most transmitted bytes belong to big and more long-lasting elephant flows, which have a large transmission time compared to the reconfiguration time. For example, Roy et al. [70] observed 90% bytes flow in elephant flows, and Griner et al. [41] give examples where a 500 MB flow, whose transmission time is 100 ms, with the reconfiguration time being 15 ms, while many other empirical studies show similar results, e.g., Mellette et al. [61], Venkatakrishnan et al. [80]. Based on these practical observations, we introduce a factor $\theta \in [0, 1]$ to indicate the ratio of reconfiguration time to the interval of a demand in our evaluations and we broadly discuss the results for $\theta = 0$ and $\theta = 0.05$ respectively in Section 5, which reveals the robustness of our algorithm under the interference of reconfiguration time.

Notwithstanding, in general, the problem of how to deal with the non-availability of optical links during reconfiguration is still an open research problem, as discussed by Nance Hall et al. [43, Section 6]: "*Ideally, we want a reconfigurable link to exist* before *the traffic appears*", with the additional challenge of these changes being consistent [35, 49].

5 EVALUATIONS

In order to study the performance of our algorithms under realistic workloads, we conducted extensive experiments with a simulator, which we will release together with this article (as open source code). In particular, we benchmark our hybrid switch algorithms against several state-of-the-art maximum matching and greedy baselines, considering a spectrum of packet traces on hybrid switch topologies as in Figure 2. We first describe our methodology in Section 5.1 and then discuss our results in Section 5.2. To facilitate reproducibility, our source code is available at https://gitlab.cs.univie.ac.at/ct-papers/2021-tompecs-load-optimization.

5.1 Methodology

Comparison with related work. We consider the following approaches from related work, used in multiple state-of-the-art articles [43], as described next, and implemented the corresponding algorithms for comparison.

— First, we compare our *h*ybrid *s*witch *n*etwork algorithms (denoted by *HSN-US/SN*) with a *Maximum Weight Matching* algorithm as a baseline, where routing occurs either on direct reconfigurable links or via the central packet switch. The matching algorithm is employed by many state-of-the-art systems [61, Table 1], also recently e.g., in *Chopin* [71]. Its use was

[7] This is not always the case for wide-area networks [30].

spearheaded by *Helios* and *c-Through* [29, 82] and it is also optimal w.r.t. the average weighted path length [33] in such a routing model.[8]
— Second, we also compare to a *Greedy* approach used by, e.g., Halperin et al. [44] and Zheng et al. [91]. For the link e that currently has the highest load, we check for the largest flow that can be rerouted on a direct connection, and offload it from the electrically switched network parts. This process is iterated until the load cannot be reduced further, where different links e can be chosen in each iteration.

Hence, in the following plots, the approaches that correspond to related work are labeled as *Max Weight Matching* and *Greedy*, respectively. Lastly, we additionally plot the maximum load on the network before any reconfiguration was applied (labeled as *Oblivious*).

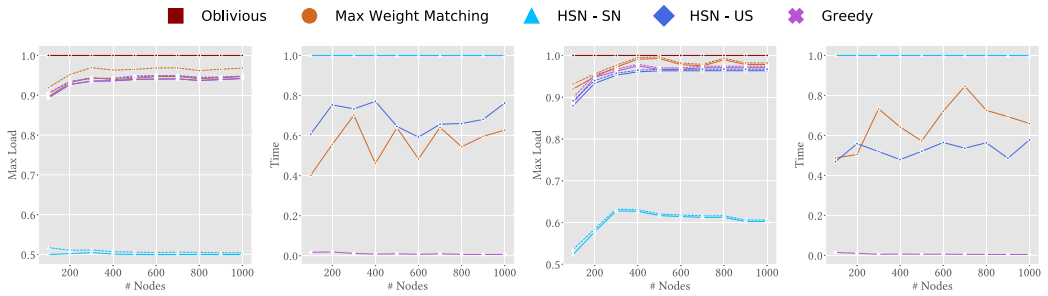
Traffic workloads. It is known that traffic traces in different networks and running different applications can differ significantly [7, 13, 37, 51, 70]. Thus, we collected a number of real-world and synthetic datasets from which we generate traffic matrices to evaluate and compare the performance of our algorithms. In particular:

— **Data center traces:** We consider two data center workloads, based on traces made available by Facebook [28, 70, 90]. The first workload features traces from a cluster running the batch-processing application Hadoop. The second one consists of traces from a cluster running SQL databases. Both workloads differ heavily in their communication patterns and the overall network load. Hence, the structural and temporal patterns of the workloads are quite different [7].
— **HPC traces:** We further consider a high performance computing workload, obtained from the CESAR backbone [2] The workload consists of a collection of MPI traffic, which was collected while running the application Nekbone. The application solves poison equations using the conjugate gradient method.
— **Synthetic traces:** The synthetic pFabric traces are frequently considered as benchmarks in scientific evaluations [6]. In a nutshell, workloads arrive according to a Poisson process, are embedded in a data center context, and follow a random communication pattern between subsets of nodes. In order to generate traffic traces and produce demand matrices, we use the NS2 simulation script we obtained from the authors, using the parameter $p = 0.5$.

In more detail, for each simulation setting, e.g., $100 - 1{,}000$ or $1{,}000 - 3{,}000$ nodes, we pre-fetch a sequence of requests and keep it in memory. For example, to observe 3,000 distinct nodes in the case of Facebook's data center traffic, we have to fetch a much larger traffic sequence, than in the case of 1,000 distinct nodes. Furthermore, to ensure fairness, the fetched traffic sequence does not stop at the last node discovered, but rather goes slightly beyond that, to allow the last discovered node to eventually be observed a few times in subsequent requests. Subsequently, depending on the current amount of nodes n, we only use the requests from the fetched sequence, where traffic occurs between those n nodes. Hence, the computational workload for, e.g., 1,000 nodes is higher in the setting of 1,000–3,000 nodes in comparison to the setting of 1,000–3,000 nodes.

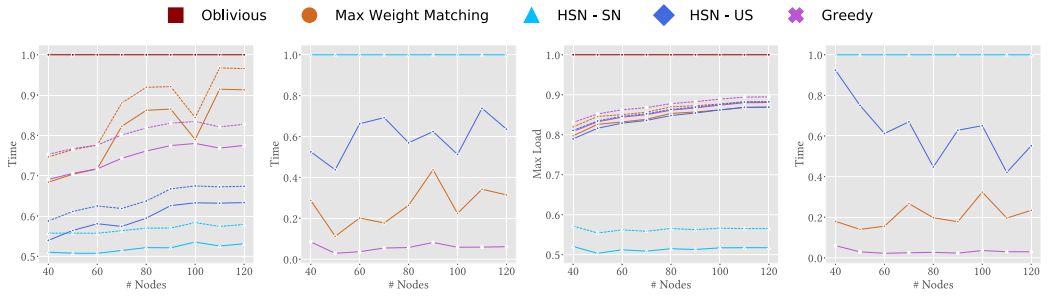
Reconfiguration delay. In order to model the reconfiguration delays of optical circuit switches, our approach is the following. We account for the reconfiguration delay by introducing a penalty parameter $\theta \in [0, 1]$, which denotes the percentage of time per traffic sequence a switch needs for reconfiguration. We first compute the optimized load of the network as if no reconfiguration delay

[8]Note that a maximum matching algorithm is not optimal regarding path lengths in all topologies. However, when the distances between all nodes are identical in the static network part, a standard maximum matching approach is optimal in hybrid switch networks w.r.t. weighted path length [33].



(a) Database: Max Load (b) Database: Runtime (c) Hadoop: Max Load (d) Hadoop: Runtime

Fig. 4. Algorithmic comparison of the maximum load and runtime for different Facebook clusters. W.r.t. max. load, Greedy, Max Weight Matching, and HSN-SN perform at very similar levels.



(a) pFabric: Max Load (b) pFabric: Runtime (c) HPC: Max Load (d) HPC: Runtime

Fig. 5. Algorithmic comparison of the maximum load and runtime for pFabric and HPC traces. W.r.t. max. load, Greedy, Max Weight Matching, and HSN-SN perform at very similar levels for HPC.

applies. Then, we query the network for the optical link load and redistribute ($load * \theta$) amount of bytes from the optical link to the electrical links. Finally, we query the network again for the maximum load.

Experimental setup. All considered topologies, ranging from 40 to 3,000 nodes,[9] employ hybrid switch networks as in Figure 2(b).[10] We repeat each setting by running it 5 times and display the averaged results, normalizing the workload in the static topology. For the runtime, we also display the averaged results, normalizing them against the results of our HSN-SN algorithm.

Our simulations were run on a machine with two Intel Xeons E5-2697V3 SR1XF with 2.6 GHz, 14 cores[11] each and a total of 128 GB RAM. The host machine was running Ubuntu 18.04.3 LTS.

We implemented the algorithms in Python (3.7.3) leveraging the NetworkX library (2.3). For the implementation of the maximum matching algorithm we used the algorithm provided by NetworkX.

5.2 Results and Discussion

We report on the main results obtained in our simulations based on the different datasets. Figures 4 and 6 summarize our evaluation results in terms of load and runtime for the Facebook traces; Figure 5 shows the corresponding results for the HPC and pFabric traces.

[9]See Alistarh et al. [5] w.r.t. the feasibility of 1,000 port optical switches in data centers.
[10]In other words, we assume that the static networks can be abstracted as trees of depth one, due to them being, e.g., non-blocking, such as for fat-trees or Clos topologies in general.
[11]However, each algorithm only utilized a single core.

Potential for load optimization. All algorithms significantly improve the load over the Oblivious baseline and provide relatively stable benefits throughout all scenarios investigated. We evaluate all algorithms with and without a reconfiguration delay, where the dashed lines in the maximum load plots correspond to the results achieved with a reconfiguration delay applied. The reconfiguration delay penalty θ is set to 0.05 in all experiments. Hence, 5% of the load on an optical link is redistributed to the electrical links to account for the reconfiguration delay.

Among these algorithms, the HSN algorithms typically clearly outperform the others.

More specifically, for the database (Figures 4(a) and 6(a)) clusters, the reduction in the maximum load provided by the HSN-SN algorithm is almost a factor of two throughout the spectrum.

For the Hadoop clusters (Figures 4(c) and 6(b)), the performance of HSN-SN slightly decreases, but still achieves $\approx$ 60% of the original Oblivious load up until a network size of 1,000 and then stays stable at $\approx$ 70% beyond. The three remaining algorithms (Greedy, Max. Weight Matching, and our HSN-US) achieve nearly identical values, with Greedy and HSN-US being slightly better. Above 1,000 nodes, we can observe that their capability to further reduce the load seems to be quite restricted. In some Hadoop workload instances, Max. Weight Matching achieves no or only minimal load reduction results, matching up Lemma 4.9 to practice. Notwithstanding, they always perform significantly worse than HSN-SN, resulting in a comparatively load-increase of $\approx$ 60%.

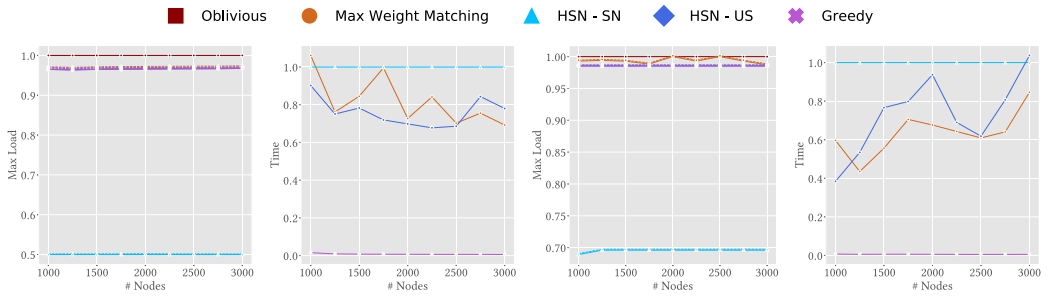
Regarding the HPC traces, we can observe similar results as in the Database Cluster, in terms of maximum load reduction. Also for the pFabric traces, our HSN-US algorithm achieves a lower maximum load compared to the Greedy or Max. Weight Matching. Here, the variance is slightly higher than in the other experiments; this matches empirical observations on the complexity of the traces produced by these synthetic traces [7].

In regard to the maximum load reduction, we conclude that our HSN-SN algorithm is quite stable w.r.t. to the number of nodes in the network. In contrast to that, Max. Weight Matching and the Greedy algorithm asymptotically approach the maximum load of the unconfigured network.

With respect to the results achieved while using the reconfiguration delay penalty θ, we can observe that the maximum link load is slightly higher for all algorithms. However, the simulations show that the reconfiguration delay penalty has a larger impact on our HSN-SN algorithm. The reason for this is that the HSN-SN algorithm is capable of distributing the traffic load more equally between the optical and electrical links. Therefore, redistributing 5% ($\theta = 0.05$) of the load from the optical links to the electrical links results in an approximately 5% increase of the load on the electrical link, which then carries the maximum load. Compared to that, the other algorithms fail to offload a significant amount of traffic to the optical links. Hence, the reconfiguration delay has a minor influence on the maximum link load because the electrical links already carry the vast amount of traffic.

Runtime performance. The best runtime is generally achieved by the Greedy algorithm, due to its early termination when no link can be added anymore. Our experiments show that in the case of the Greedy algorithm, this is unfortunately happening very early on. Regarding the runtime of the Max. Weight Matching, we want to emphasize that the algorithm is unaware of the underlying problem of reducing the maximum link load. Therefore, a lot of runtime is actually wasted without achieving any further load reduction. Hence, in some cases, e.g., in the larger Facebook clusters, Max. Weight Matching is even slower than HSN-SN. In comparison to Max. Weight Matching, our HSN-US has a similar runtime, while spending all of it searching for the best load reduction matching.

HSN-US is consistently faster than HSN-SN, and the latter features quite a high variance in runtime. Notwithstanding, HSN-US has the benefit of only routing along single paths, which can be beneficial for performance metrics beyond load [72, 87]. On the other hand, such issues can also be alleviated with specialised multipath procotols [23, 68, 83]. Still, in some cases and specific



(a) Database: Max Load (b) Database: Runtime (c) Hadoop: Max Load (d) Hadoop: Runtime

Fig. 6. Algorithmic comparison of the maximum load and runtime for different larger Facebook clusters. W.r.t. max. load, Greedy, Max Weight Matching, and HSN-SN perform at very similar levels.

workloads, the routing of related demands becomes easier in the SN model. Hence HSN-SN can even be slightly faster than HSN-US, such as for the Hadoop cluster at 3,000 nodes, due to the fact that the underlying matching problem is identical for both HSN-US/-SN.

Summary. While all algorithms provide load reductions, the extent of these optimizations and the required runtime differ significantly. Our results suggest that the load optimizations provided by HSN-US might prove beneficial over other segregated routing strategies, particularly because of its low runtime which is comparable to that of the Max. Weight Matching. We conclude that when considering both potential load reduction and runtime, HSN-SN provides a better tradeoff than HSN-US.

6 RELATED WORK

Most related work on flow routing in data center networks focuses on non-reconfigurable topologies [64]. That said, many recent works design and evaluate reconfigurable topologies e.g., [17, 29, 37, 44, 55, 56, 59–61, 67, 80–82, 85, 86], often showing significant performance gains over static topologies and proving real-world viability. However, the algorithmic complexity of reconfigurable data center networks is mostly unstudied [34], and many fundamental questions remain open [11].

Scheduling traffic matrices with specific skew were investigated in [56, 57, 67, 80], but performance guarantees were only obtained by Venkatakrishnan et al. [80] due to leveraging submodularity, a condition that does not hold in our setting. Similarly, Avin et al. [8–10] investigate traffic matrices with low entropy, but they require scalable constant reconfigurable degrees and are oblivious to hybrid networks, as in [16, 65], and thus do not translate to the herein considered model.

The idea of leveraging good connectivity in data center contexts arose from utilizing random graphs [75], and later extended into deterministic versions [22, 52, 77]. Xia et al. [86] used this idea to heuristically switch between random graphs and Clos topologies, depending on the traffic pattern, whereas Mellette et al. [60] incorporate it to improve their Rotornet [61] approach: If a flow cannot be delayed respectively be buffered, it gets sent along a short route. Both works of Mellette et al. also have the benefit that their reconfigurations are oblivious to the current traffic pattern, but hence also depend on the same for the resulting performance.

One of the notable works that does not rely on centralized computation is ProjecToR by Ghobadi et al. [37], which instead performs a distributed matching protocol reminiscent to the idea of stable matchings [1]. In their setting, they obtain a $(2 + \varepsilon)$ approximation for the weighted latency objective but do not consider load.

The algorithmic complexity of weighted latency was also considered in [32, 33], where already basic topologies and settings turned out to be intractable. On the other hand, finding a single shortest path in a partially reconfigured network can be done efficiently, and hence yields well performing heuristics [31]. Moreover, some routing models can even be solved optimally. Notwithstanding, it is unclear how to transfer these results to a load-optimization setting: in topologies with unfavorable betweenness centrality, shortest path routing can overload popular links with high load.

Load-optimization in reconfigurable data centers was recently studied by Yang et al. [87], who investigated the impact of wireless interference on cross-layer optimization. Different wireless links are modeled as a conflict graph, where the task is to find sufficiently good independent (link) sets, in order to provide an interference-free reconfiguration. We see our work as orthogonal, as we only consider inherently interference-free technologies, and as thus it would be interesting to leverage their results in future work.

Another interesting line of work is by Zheng et al. [91], who study how to enhance the design of Diamond, BCube, and VL2 network topologies with small reconfigurable switches, inspired by Flat-Tree [86]. They target maximum link load as well, and present intractability results on general graphs, although these results do not transfer to specific data center topologies or trees, respectively. Different routing models are not analyzed. Moreover, they propose to reconfigure the network with a greedy algorithm, which however does not come with formal performance guarantees. In evaluations of small network sizes, their combination of greedy algorithm and enhanced network design reduces the maximum load by 12% on average. We see similar greedy algorithm behavior in our evaluations, where however the greedy algorithm performance decreases to just a few percent of load improvement as the network size grows.

That being said, even though our work is mostly motivated by technologies emerging in data center networks, it also applies to other reconfigurable technologies, as long as they fulfill our model properties. Fundamentally different however are reconfigurable optical wide-area networks, as therein the fiber connectivity is fixed. Hence capacities can be adjusted and alternative failover paths provided, leading to improvements in the scheduling of bulk-transfers [21, 48, 49, 58] and reliability concerns [39, 42, 74, 92].

7 CONCLUSION

We investigated load minimization in reconfigurable hybrid networks, leveraging the flexibility of emerging programmable physical layers. To this end, we investigated the underlying problem complexity, unveiling that already tree topologies of small height induce intractability for a multitude of routing models, and that one cannot hope for general approximability via submodularity techniques. Notwithstanding, we showed that hybrid switch networks, and in turn, non-blocking data center interconnects, can be optimized efficiently. Trace-driven simulations show that our hybrid switch algorithms significantly outperform a state-of-the-art maximum matching baseline, but also greedy algorithms.

APPENDIX

A DEFERRED PROOFS AND ALGORITHMS

A.1 Proof of Lemma 4.3 for $\tau = \mathbf{SN}$

Lemma A.1. *Given a reconfigured network $N(M)$, which is a triangle on nodes $V = \{a, b, c\}$ with the only configured link $\{a, b\} \in \mathcal{E}$, then for demands D, a load-optimization flow f_{opt} in $N(M)$ can be computed in constant time by Algorithm 5 when $\tau = SN$.*

Proof. When τ = SN, any two distinct demands in D in the triangle $\{a, b, c\}$ are called related if they share the same source or sink. Let f be an arbitrary flow serving D under τ = SN. For any two related demands, e.g., $D(a, b)$ and $D(a, c)$, W.L.O.G., we assume $D(a, b)$ sending a flow of size $\beta > 0$ along (a, c, b) and $D(a, c)$ sending a flow of size $\alpha > 0$ along (a, b, c) in f; and remaining of $D(a, b)$ and $D(a, c)$ are only sent on directed links (a, b) and (a, c), respectively, in f. We call such a routing as interfering for these two related demands. W.L.O.G, we also assume $\beta \geq \alpha$. The interfering between $D(a, b)$ and $D(a, c)$ in f can be canceled by redirecting a flow of size α of $D(a, c)$ from its indirect path (a, b, c) to its shortcut (a, c), while forcing $D(a, b)$ only sending a flow of size $\beta - \alpha$ along (a, c, b). Clearly, the cancellation would not increase the maximum load of f. Thus, there must be a load-optimization flow f^* serving D such that no interfering occurs between any two related demands, otherwise we can do the interfering cancellation in f^*.

Now, we need to find the load-optimization flow f^*. Given a triangle N and demands, we will prove that Algorithm 5 can find f^* in constant time. Clearly, Algorithm 5 terminates in constant time since the number of demands is at most 6. It is clear that the returned flow f_{opt} is an interfering-free flow since when a demand $D(u, v)$ is marked split, all its related demands are rejected for being further splitted. Given an upper-bound μ, our algorithm guarantees that all directed links have loads no more than μ. Now, we just need to prove that μ found in Algorithm 4 is minimum. We assume that $\mu' < \mu$ is actually the minimized maximum load. Each demand marked as split in Algorithm 5: $\forall D(u, v) \in D_S$ must send a flow of size $D(u, v) - \mu' C(u, v)$ to its indirect path, where $D(u, v) - \mu' C(u, v) > D(u, v) - \mu C(u, v)$, otherwise, some links would have loads more than μ'. Due to the interfering-free requirement, each demand in $D \setminus D_S$ cannot send its flow to its indirect path. W.L.O.G, let $D(p, q)$ be the unsplit demand in $D \setminus D_S$, which has the maximum load μ in S_μ in Algorithm 4. Since the related demands of $D(p, q)$, which are marked as split, need to send more flows to their indirect paths, where (p, q) is included. Then the load on the link (p, q) will be larger than μ, which contradicts the assumption. □

ALGORITHM 4: Determine Optimal Load

Input : A Triangle N on three nodes V, demands D, a set of split demands D_S;
Output : a load-otimization flow f_{opt} for N;

Define a set $S_\mu = \emptyset$ and a variable $\mu \in \mathbb{R}^+$;
for *each unsplit demand* $D(u, v) \in D \setminus D_S$ **do**
  Let $D' \subseteq D_s$ be "split" demads related to $D(u, v)$;
  $D(u, v)$ is only sent on (u, v) in f_{opt};
  If $|D'| = 2$, let $D' = \{D(u, d), D(d, v)\}$;
  compute $\mu = \frac{D(u,v)+D(u,d)+D(d,v)}{C(u,v)+C(u,d)+C(d,v)}$ and $S_\mu = S_\mu \cup \{\mu\}$;
  If $|D'| \leq 1$, similar computation of μ;
$\mu = \max\{\mu' : \mu' \in S_\mu\}$;
for *each split demand* $D(p, q) \in D_S$ **do**
  $D(p, q)$ send a flow of size $\mu \cdot C(p, q)$on (p, q) and remaning flow on (p, k, q) in f_{opt}, where $k \in V$ and $k \notin \{q, p\}$;
return f_{opt};

ALGORITHM 5: Triangle Optimization When $\tau = SN$

Input : A Triangle $N = (V, E, \mathcal{E}, C)$ with nodes $V = \{a, b, c\}$, demands D, and the configured link (bidirected) $\{a, b\} \in \mathcal{E}$;

Output : a load-otimization flow f_{opt} for N;

Define a set $D_S = \emptyset$ and values $\forall D(u, v) \in D : \alpha_{uv} = \frac{D(u,v)}{C(u,v)}$;
Let any two demands $D(u, v)$ and $D(p, q)$, where $u, v, p, q \in V$, $u \neq v$, and $p \neq q$, be related if either $u = p$ or $v = q$;
Let $D(u, v)$ be the demand in D with the highest value α_{uv};
if *a demand* $D(p, q)$ *is related to* $D(u, v)$ *and* $\alpha_{pq} = \alpha_{uv}$ **then**
    return f_{opt} = Algorithm 4(N, D, D_S);
Mark the demand $D(u, v)$ "split" and $D_S = D_S \cup \{D(u, v)\}$;
Set $D_1 = \{D(u, v), D(u, d), D(d, v)\}$, where $d = V \setminus \{u, v\}$;
Let $D(p, q)$ be a demand in $D_2 := D \setminus D_1$ with the highest α_{pq};
if $\exists D(i, j) \in D_1$ *is unsplit and has* $\alpha_{ij} \geq \alpha_{pq}$ *for* $D(p, q)$ **then**
    return f_{opt} = Algorithm 4(N, D, D_S);
if $\exists D(i, j) \in D_2$ *is related to* $D(p, q)$ *and has* $\alpha_{ij} \geq \alpha_{pq}$ **then**
    return f_{opt} = Algorithm 4(N, D, D_S);
Mark the demand $D(p, q)$ "split" and $D_S = D_S \cup \{D(p, q)\}$;
Set $D_3 = \{D(p, q), D(p, d), D(d, q)\}$, where $d = V \setminus \{q, p\}$;
if $D(f, g) := D_2 \setminus D_3 \neq \emptyset$, *where* $f, g \in V$ *and* $f \neq g$ **then**
    if $\nexists D(i, j) \in D \setminus D(f, g)$ *is unsplit and has* $\alpha_{ij} \geq \alpha_{fg}$ **then**
        Mark $D(f, g)$ "split" and $D_S = D_S \cup \{D(f, g)\}$;
return f_{opt} = Algorithm 4(N, D, D_S);

REFERENCES

[1] Nobel Prize Outreach AB. 2012. *The Prize in Economic Sciences 2012.* Retrieved from https://www.nobelprize.org/prizes/economic-sciences/2012/summary/. Accessed 11-11-2021.

[2] US DOE. 2016. *Characterization of the DOE Mini-apps.* portal.nersc.gov/project/CAL/doe-miniapps.htm.

[3] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1993. *Network Flows - Theory, Algorithms and Applications.* Prentice Hall.

[4] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *Proceedings of the SIGCOMM.*

[5] Dan Alistarh, Hitesh Ballani, Paolo Costa, Adam C. Funnell, Joshua Benjamin, Philip M. Watts, and Benn Thomsen. 2015. A high-radix, low-latency optical switch for data centers. *Computer Communication Review* 45, 5 (2015), 367–368.

[6] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. pFabric: Minimal near-optimal datacenter transport. In *Proceedings of the SIGCOMM.* ACM.

[7] Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the complexity of traffic traces and implications. In *Proceedings of the ACM SIGMETRICS.*

[8] Chen Avin, Alexandr Hercules, Andreas Loukas, and Stefan Schmid. 2018. *rDAN*: Toward robust demand-aware network designs. *Information Processing Letters* 133 (2018), 5–9.

[9] Chen Avin, Kaushik Mondal, and Stefan Schmid. 2017. Demand-aware network designs of bounded degree. In *Proceedings of the DISC.*

[10] Chen Avin, Kaushik Mondal, and Stefan Schmid. 2019. Demand-aware network design with minimal congestion and route lengths. In *Proceedings of the INFOCOM.* IEEE.

[11] Chen Avin and Stefan Schmid. 2018. Toward demand-aware networking: A theory for self-adjusting networks. *Computer Communication Review* 48, 5 (2018), 31–40.

[12] Navid Hamed Azimi, Zafar Ayyub Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. 2014. FireFly: A reconfigurable wireless data center fabric using free-space optics. In *Proceedings of the SIGCOMM.* ACM.

[13] Theophilus Benson, Aditya Akella, and David A. Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the Internet Measurement Conference*. ACM.
[14] Calient. 2018. Edge 640 Optical Circuit Switch. Retrieved from https://www.calient.net/products/edge640-optical-circuit-switch/. Accessed 11-11-2021.
[15] Jiaxin Cao, Rui Xia, Pengkun Yang, Chuanxiong Guo, Guohan Lu, Lihua Yuan, Yixin Zheng, Haitao Wu, Yongqiang Xiong, and David A. Maltz. 2013. Per-packet load-balanced, low-latency routing for clos-based data center networks. In *Proceedings of the CoNEXT*. ACM.
[16] Esra Ceylan, Klaus-Tycho Foerster, Stefan Schmid, and Katsiaryna Zaitsava. 2021. Demand-aware plane spanners of bounded degree. In *Proceedings of the Networking*. IEEE, 1–9.
[17] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. 2014. OSA: An optical switching architecture for data center networks with unprecedented flexibility. *IEEE/ACM Transactions on Networking* 22, 2 (2014), 498–511.
[18] Charles Clos. 1953. A study of non-blocking switching networks. *Bell System Technical Journal* 32, 2 (1953), 406–424.
[19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition* (3rd. ed.). The MIT Press.
[20] Wenkai Dai, Michael Dinitz, Klaus-Tycho Foerster, and Stefan Schmid. 2022. Brief announcement: Minimizing congestion in hybrid demand-aware network topologies. In *Proceedings of the 36th International Symposium on Distributed Computing*. Christian Scheideler (Ed.), Leibniz International Proceedings in Informatics, Vol. 246, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 42:1–42:3. DOI: https://doi.org/10.4230/LIPIcs.DISC.2022.42
[21] Michael Dinitz and Benjamin Moseley. 2020. Scheduling for weighted flow and completion times in reconfigurable networks. In *Proceedings of the INFOCOM*. IEEE.
[22] Michael Dinitz, Michael Schapira, and Asaf Valadarsky. 2017. Explicit expanding expanders. *Algorithmica* 78, 4 (2017), 1225–1245.
[23] Advait Abhay Dixit, Pawan Prakash, Y. Charlie Hu, and Ramana Rao Kompella. 2013. On the impact of packet spraying in data center networks. In *Proceedings of the INFOCOM*. IEEE.
[24] Ran Duan and Seth Pettie. 2014. Linear-time approximation for maximum weight matching. *Journal of the ACM* 61, 1 (2014), 1:1–1:23.
[25] Nick G. Duffield, Pawan Goyal, Albert G. Greenberg, Partho Pratim Mishra, K. K. Ramakrishnan, and Jacobus E. van der Merwe. 1999. A flexible model for resource management in virtual private networks. In *Proceedings of the SIGCOMM*. ACM, 95–108.
[26] Jack Edmonds. 1965. Paths, trees, and flowers. *Canadian Journal of Mathematics* 17 (1965), 449–467.
[27] Tom Empson and Scarlet Schwiderski-Grosche. 2019. Optics for the Cloud Research Alliance Establishes Collaborative Research Approach to Improving Cloud Technology. Retrieved from https://www.microsoft.com/en-us/research/blog/optics-for-the-cloud-research-alliance-establishes-collaborative-research-approach-to-improving-cloud-technology/. Accessed 11-11-2021.
[28] facebook. 2018. Facebook Network Analytics Data Sharing. Retrieved from https://www.facebook.com/groups/1144031739005495/. Accessed 11-11-2021.
[29] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2010. Helios: A hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of the SIGCOMM*. ACM.
[30] Thomas Fenz, Klaus-Tycho Foerster, and Stefan Schmid. 2021. On efficient oblivious wavelength assignments for programmable wide-area topologies. In *Proceedings of the ANCS*. ACM, 38–51.
[31] Thomas Fenz, Klaus-Tycho Foerster, Stefan Schmid, and Anaïs Villedieu. 2020. Efficient non-segregated routing for reconfigurable demand-aware networks. *Computer Communications* 164 (2020), 138–147.
[32] Klaus-Tycho Foerster, Maciej Pacut, and Stefan Schmid. 2019. On the complexity of non-segregated routing in reconfigurable data center architectures. *Computer Communication Review* 49, 2 (2019), 2–8.
[33] Klaus-T. Foerster, Manya Ghobadi, and Stefan Schmid. 2018. Characterizing the algorithmic complexity of reconfigurable data center architectures. In *Proceedings of the ANCS*. ACM.
[34] Klaus-T. Foerster and Stefan Schmid. 2019. Survey of reconfigurable data center networks: Enablers, algorithms, complexity. *SIGACT News* 50, 2 (2019), 62–79.
[35] Klaus-Tycho Foerster, Stefan Schmid, and Stefano Vissicchio. 2019. Survey of consistent software-defined network updates. *IEEE Communications Surveys and Tutorials* 21, 2 (2019), 1435–1461.
[36] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York.

[37] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil R. Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel C. Kilper. 2016. ProjecToR: Agile reconfigurable data center interconnect. In *Proceedings of the SIGCOMM*. ACM.

[38] Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab Mirrokni. 2009. Approximating submodular functions everywhere. In *Proceedings of the SODA*.

[39] Jennifer Gossels, Gagan Choudhury, and Jennifer Rexford. 2019. Robust network design for IP/optical backbones. *Journal of Optical Communications and Networking* 11, 8 (2019), 478–490.

[40] Albert G. Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2011. VL2: A scalable and flexible data center network. *Communications of the ACM* 54, 3 (2011), 95–104.

[41] Chen Griner, Johannes Zerwas, Andreas Blenk, Manya Ghobadi, Stefan Schmid, and Chen Avin. 2021. Cerberus: The power of choices in datacenter topology design - A throughput perspective. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3 (2021), 33 pages. DOI: https://doi.org/10.1145/3491050

[42] Matthew Nance Hall, Paul Barford, Klaus-Tycho Foerster, Manya Ghobadi, William Jensen, and Ramakrishnan Durairajan. 2021. Are WANs ready for optical topology programming?. In *Proceedings of the OptSys@SIGCOMM*. ACM, 28–33.

[43] Matthew Nance Hall, Klaus-Tycho Foerster, Stefan Schmid, and Ramakrishnan Durairajan. 2021. A survey of reconfigurable optical networks. *Optical Switching and Networking* 41 (2021), 100621.

[44] Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. 2011. Augmenting data center networks with multi-gigabit wireless links. In *Proceedings of the SIGCOMM*. ACM.

[45] Abdelbaset S. Hamza, Jitender S. Deogun, and Dennis R. Alexander. 2016. Wireless communication in data centers: A survey. *IEEE Communications Surveys and Tutorials* 18, 3 (2016), 1572–1595.

[46] Kai Han, Zhiming Hu, Jun Luo, and Liu Xiang. 2015. RUSH: Routing and scheduling for hybrid data center networks. In *Proceedings of the INFOCOM*. IEEE.

[47] Jeff Hecht. 2001. Many Approaches Taken for All-optical Switching. Retrieved from https://www.laserfocusworld.com/optics/article/16556781/many-approaches-taken-for-alloptical-switching. Accessed 11-11-2021.

[48] Su Jia, Xin Jin, Golnaz Ghasemiesfeh, Jiaxin Ding, and Jie Gao. 2017. Competitive analysis for online scheduling in software-defined optical WAN. In *Proceedings of the INFOCOM*. IEEE.

[49] Xin Jin, Yiran Li, Da Wei, Siming Li, Jie Gao, Lei Xu, Guangzhi Li, Wei Xu, and Jennifer Rexford. 2016. Optimizing bulk transfers with software-defined optical WAN. In *Proceedings of the SIGCOMM*. ACM, 87–100.

[50] Christoforos Kachris and Ioannis Tomkos. 2012. A survey on optical interconnects for data centers. *IEEE Communications Surveys and Tutorials* 14, 4 (2012), 1021–1036.

[51] Srikanth Kandula, Sudipta Sengupta, Albert G. Greenberg, Parveen Patel, and Ronnie Chaiken. 2009. The nature of data center traffic: Measurements and analysis. In *Proceedings of the Internet Measurement Conference*. ACM.

[52] Simon Kassing, Asaf Valadarsky, Gal Shahaf, Michael Schapira, and Ankit Singla. 2017. Beyond fat-trees without antennae, mirrors, and disco-balls. In *Proceedings of the SIGCOMM*. ACM.

[53] Brian Lebiednik, Aman Mangal, and Niharika Tiwari. 2016. A survey and evaluation of data center network topologies. *CoRR abs/1605.01701* (2016). Retrieved from http://arxiv.org/abs/1605.01701

[54] Charles E. Leiserson. 1985. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers* 34, 10 (1985), 892–901.

[55] He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter. 2014. Circuit switching under the radar with REACToR. In *Proceedings of the NSDI*. USENIX Association.

[56] He Liu, Matthew K. Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M. Voelker, David G. Andersen, Michael Kaminsky, George Porter, and Alex C. Snoeren. 2015. Scheduling techniques for hybrid circuit/packet networks. In *Proceedings of the CoNEXT*. ACM.

[57] Ariel Livshits and Shay Vargaftik. 2018. LUMOS: A fast and efficient optical circuit switch scheduling technique. *IEEE Communications Letters* 22, 10 (2018), 2028–2031.

[58] Long Luo, Klaus-Tycho Foerster, Stefan Schmid, and Hongfang Yu. 2020. Deadline-aware multicast transfers in software-defined optical wide-area networks. *IEEE Journal on Selected Areas in Communications* 38, 7 (2020), 1584–1599.

[59] Long Luo, Klaus-Tycho Foerster, Stefan Schmid, and Hongfang Yu. 2020. SplitCast: Optimizing multicast flows in reconfigurable datacenter networks. In *Proceedings of the INFOCOM*. IEEE.

[60] William M. Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C. Snoeren, and George Porter. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *Proceedings of the NSDI*. https://dblp.org/rec/conf/nsdi/MelletteDGMSP20.html?view=bibtex.

[61] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. 2017. RotorNet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the SIGCOMM*. ACM.
[62] William M. Mellette, Alex C. Snoeren, and George Porter. 2016. P-FatTree: A multi-channel datacenter network topology. In *Proceedings of the HotNets*. ACM.
[63] Cisco Visual Networking. 2016. Cisco global cloud index: Forecast and methodology, 2015-2020. white paper. *Cisco Public, San Jose* (2016), 1–29.
[64] Mohammad Noormohammadpour and Cauligi S. Raghavendra. 2018. Datacenter traffic control: Understanding techniques and tradeoffs. *IEEE Communications Surveys and Tutorials* 20, 2 (2018), 1492–1525.
[65] Maciej Pacut, Wenkai Dai, Alexandre Labbe, Klaus-Tycho Foerster, and Stefan Schmid. 2021. Improved scalability of demand-aware datacenter topologies with minimal route lengths and congestion. *Performance Evaluation* 152 (2021), 102238.
[66] Polatis. 2019. Series 6000n Network Optical Matrix Switch. Retrieved from https://www.hubersuhner.com/en/documents-repository/technologies/pdf/data-sheets-optical-switches/polatis-series-6000n. Accessed 11-11-2021.
[67] George Porter, Richard D. Strong, Nathan Farrington, Alex Forencich, Pang-Chen Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2013. Integrating microsecond circuit switching into the data center. In *Proceedings of the SIGCOMM*. ACM.
[68] Costin Raiciu, Sébastien Barré, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. 2011. Improving datacenter performance and robustness with multipath TCP. In *Proceedings of the SIGCOMM*. ACM.
[69] Yongmao Ren, Yu Zhao, Pei Liu, Ke Dou, and Jun Li. 2014. A survey on TCP Incast in data center networks. *International Journal of Communication Systems* 27, 8 (2014), 1160–1172.
[70] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the social network's (Datacenter) network. In *SIGCOMM*, ACM, 123–137.
[71] Neta Rozen-Schiff, Klaus-Tycho Foerster, Stefan Schmid, and David Hay. 2022. Chopin: Combining distributed and centralized schedulers for self-adjusting datacenter networks. In *Proceedings of the OPODIS (LIPIcs'22)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.
[72] Siddhartha Sen, David Shue, Sunghwan Ihm, and Michael J. Freedman. 2013. Scalable, optimal flow routing in datacenters via local link balancing. In *Proceedings of the CoNEXT*. ACM.
[73] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Hong Liu, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2016. Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network. *Communications of the ACM* 59, 9 (2016), 88–97.
[74] Rachee Singh, Manya Ghobadi, Klaus-T. Foerster, Mark Filer, and Phillipa Gill. 2018. RADWAN: Rate adaptive wide area network. In *Proceedings of the SIGCOMM*. ACM.
[75] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. 2012. Jellyfish: Networking data centers randomly. In *Proceedings of the NSDI*. USENIX.
[76] Ashwin Sridharan, Roch Guérin, and Christophe Diot. 2005. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Transactions on Networking* 13, 2 (2005), 234–247.
[77] Asaf Valadarsky, Gal Shahaf, Michael Dinitz, and Michael Schapira. 2016. Xpander: Towards optimal-performance datacenters. In *Proceedings of the CoNEXT*. ACM.
[78] Vijay V. Vazirani. 2001. *Approximation Algorithms*. Springer, Berlin.
[79] Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, and Pramod Viswanath. 2016. Costly circuits, submodular schedules and approximate Carathéodory theorems. In *Proceedings of the SIGMETRICS*. ACM.
[80] Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, and Pramod Viswanath. 2018. Costly circuits, submodular schedules and approximate Carathéodory theorems. *Queueing Systems* 88, 3–4 (2018), 311–347.
[81] Guohui Wang, David G. Andersen, Michael Kaminsky, Michael Kozuch, T. S. Eugene Ng, Konstantina Papagiannaki, Madeleine Glick, and Lily B. Mummert. 2009. Your data center is a router: The case for reconfigurable optical circuit switched paths. In *Proceedings of the HotNets*. ACM.
[82] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T. S. Eugene Ng, Michael Kozuch, and Michael P. Ryan. 2010. c-Through: Part-time optics in data centers. In *Proceedings of the SIGCOMM*. ACM.
[83] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, implementation and evaluation of congestion control for multipath TCP. In *Proceedings of the NSDI*. USENIX Association.
[84] Wenfeng Xia, Peng Zhao, Yonggang Wen, and Haiyong Xie. 2017. A survey on data center networking (DCN): Infrastructure and operations. *IEEE Communications Surveys and Tutorials* 19, 1 (2017), 640–656.
[85] Yiting Xia, T. S. Eugene Ng, and Xiaoye Steven Sun. 2015. Blast: Accelerating high-performance data analytics applications by optical multicast. In *Proceedings of the INFOCOM*.
[86] Yiting Xia, Xiaoye Steven Sun, Simbarashe Dzinamarira, Dingming Wu, Xin Sunny Huang, and T. S. Eugene Ng. 2017. A tale of two topologies: Exploring convertible data center network architectures with flat-tree. In *Proceedings of the SIGCOMM*. ACM.

[87] Zhenjie Yang, Yong Cui, Shihan Xiao, Xin Wang, Minming Li, Chuming Li, and Yadong Liu. 2019. Achieving efficient routing in reconfigurable DCNs. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 3 (2019), 47:1–47:30.

[88] Zhenjie Yang, Yong Cui, Shihan Xiao, Xin Wang, Minming Li, Chuming Li, and Yadong Liu. 2020. Achieving efficient routing in reconfigurable DCNs. In *Proceedings of the SIGMETRICS (Abstracts)*. ACM.

[89] Xin Yuan. 2011. On nonblocking folded-clos networks in computer communication environments. In *Proceedings of the IPDPS*. IEEE.

[90] James Hongyi Zeng. 2017. Data Sharing on Traffic Pattern Inside Facebook's Datacenter Network. Retrieved from https://research.fb.com/data-sharing-on-traffic-pattern-inside-facebooks-datacenter-network/. Accessed 11-11-2021.

[91] Jiaqi Zheng, Qiming Zheng, Xiaofeng Gao, and Guihai Chen. 2019. Dynamic load balancing in hybrid switching data center networks with converters. In *Proceedings of the ICPP*. ACM.

[92] Zhizhen Zhong, Manya Ghobadi, Alaa Khaddaj, Jonathan Leach, Yiting Xia, and Ying Zhang. 2021. ARROW: Restoration-aware traffic engineering. In *Proceedings of the SIGCOMM*. ACM, 560–579.

Received 30 December 2020; revised 9 December 2022; accepted 5 May 2023